NPS-IJWA-01-006

DEVELOPMENT OF A PROTOTYPE RELATIONAL DATABASE SYSTEM FOR MANAGING FLEET BATTLE EXPERIMENT DATA

Kevin Colon
Magdi Kamel
Kishore Sengupta

The Institute for Joint Warfare Analysis
Naval Postgraduate School
Monterey, California

NPS-IJWA-01-006

DEVELOPMENT OF A PROTOTYPE RELATIONAL DATABASE SYSTEM FOR MANAGING FLEET BATTLE EXPERIMENT DATA

Kevin Colon
Magdi Kamel
Kishore Sengupta

INSTITUTE FOR JOINT WARFARE ANALYSIS
NAVAL POSTGRADUATE SCHOOL
Monterey, California

RADM David R. Ellison
Superintendent

Richard Ester
Provost

This report was prepared for and funded by:
Institute for Joint Warfare Analysis, and
Joint Experimentation command and the Office of Naval Research
through the CDTEMS program

This report was prepared by:

Institute For Joint Warfare Analysis
Naval Postgraduate School
Monterey, CA

REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188

GENCY USE ONLY (Leave blank)	2. REPORT DATE June 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis

TLE AND SUBTITLE
velopment of a Prototype Relational Database System for Managing Fleet Battle periment Data

5. FUNDING
ONR# N0001400WR20307

UTHOR(S)
evin Colon, Magdi Kamel, and Kishore Sengupta

ERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ute for Joint Warfare Analysis
l Postgraduate School
terey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-IJWA-01-006

PONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
stitute for Joint Warfare Analysis,
R, and Joint Experimentation Command through the CDTEMS ogram.

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

SUPPLEMENTARY NOTES
views expressed here are those of the authors and do not reflect the official policy or position of the Department of Defense of the U.S. Government

DISTRIBUTION/AVAILABILITY STATEMENT
pproved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

ABSTRACT (Maximum 200 words.)
research develops a prototype relational database system for storing an managing Fleet Battle Experiment (FBE) data. It is the first step in tructing a knowledge-base system for such data. The objective is to create a relational database capable of generating information from past riments for analysis and lessons learned to benefit future experiments. Research methodology included literature research of application lopment methodologies and database systems, as well as observing a FBE and gathering system requirements information from personnel that plan, gure, and participate in FBEs and war games.
evelopment of the system involved designing a schema (data model) that consists of entities, attributes, and relationships of the FBE environment. data model is transaction- (event-) based and concentrates on information flow in order to categorize and store the data. These events provide the al links between the identified entities and the capability to query the system for desired information. Finally, a prototype application against the model was developed to facilitate data entry, modification, and querying.

SUBJECT TERMS
base, Database Management System, Knowledge Management

15. NUMBER OF PAGES 260

16. PRICE CODE

ECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
JNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Approved for public release; distribution is unlimited.

DEVELOPMENT OF A PRTOTYPE RELATIONAL DATABASE SYSTEM FOR MANAGING FLEET BATTLE EXPERIMENT DATA

Kevin Colón
Ensign, United States Navy
B.S. Computer Information Systems, Jacksonville University, 1999

Submitted in partial fulfillment of the
Requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
June 2000

ABSTRACT

This research develops a prototype relational database system for storing and managing Fleet Battle Experiment (FBE) data. It is the first step in constructing a knowledge-base system for such data. The objective is to create a relational database capable of generating information from past experiments for analysis and lessons learned to benefit future experiments. Research methodology included literature research of application development methodologies and database systems, as well as observing a FBE and gathering system requirements information from personnel that plan, configure, and participate in FBEs and war games.

Development of the system involved designing a schema (data model) that consists of entities, attributes, and relationships of the FBE environment. The data model is transaction- (event-) based and concentrates on information flow in order to categorize and store the data. These events provide the logical links between the identified entities and the capability to query the system for desired information. Finally, a prototype application against the data model was developed to facilitate data entry, modification, and querying.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to acknowledge the financial support of the Institute for Joint Warfare Analysis (IJWA) that enabled me to attend the Navy's Global '99 War Game at the Naval War College in Newport, Rhode Island.

I would also like to thank Shelley Gallup, Nelson J. Irvine, Rich Kimmel, Gordon Schacher, and the rest of the IJWA staff at the Naval Postgraduate School for their time and efforts in support of my thesis. Thanks also to Martha Wring for helping arrange thesis travel on such short notice.

I would like to express my appreciation to Professors Kishore Sengupta and Magdi N. Kamel for their collaboration on this project.

I would like to thank my parents, Victor and Juana Colón, for their patience, backing, and understanding. They have always been there for me and all my endeavors, providing their love and support. Thank you. I love you very much.

I. INTRODUCTION

A. BACKGROUND

In a world full of uncertainty and the possibility of conflict, our country's military personnel must be prepared to deal with any number of potential threats. Preparedness is crucial, but training for certain scenarios may be economically infeasible for and/or logistically impossible to support.

In order to prepare for such scenarios and to test war fighting doctrine and philosophy, our armed forces have developed two types of training: Fleet Battle Experiments (FBE) and War Games. A War Game is an exercise that the military and other defense organizations use in order to gain experience at making decisions in diverse scenarios. War Games typically use computer simulations, also known as Synthetic Theaters of War (STOW), but may also be designed as discussion groups for system and doctrine experts. Objectives chosen are aimed at testing particular systems, practicing specific maneuvers and tactics, providing familiarization with certain environments and situations, and/or enhancing and honing decision-making skills. War Game scenarios are chosen so they are plausible and relevant to the objectives of the game. They are most effective for testing strategic and tactical levels of warfare.

FBEs are designed as genuine experiments, not demonstrations or exercises. Each experiment has a hypothesis and specific, carefully considered measures of effectiveness. Products from the FBEs include: new doctrine; new insight into technology in an operational environment; identification of new required operational capabilities; identification of new acquisition requirements; ideas for further warfare concepts; and ideas for further experimentation. FBEs are useful at testing all levels of an operation from strategic to operational. [Ref. 11]

The information and knowledge that is produced during the war game is vast and complex. The data produced is of varying types, recorded in dissimilar ways, and is related in a complex manner. Experiential knowledge from an FBE is carried away by

the participants and is rapidly dispersed by personnel assignments and tasking. This information would be of great benefit to future scenario design and to provide decision support to the participants. Unfortunately, it is too diverse to be appropriately collected and stored. Consequently, there is currently no appropriate approach or mechanism to systematically capture, retrieve, and analyze the data collected from an FBE.

Currently, several disparate systems are used to manage data for a FBE. For example, Land Attack Weapons System (LAWS) and Global Intelligence, Surveillance, and Reconnaissance System (GISRS) provide weapon and sensor node data management as well as the data repositories for these objects. This collection of heterogenous systems causes problems in the integration of the data for the purpose of querying and analysis. Additionally, data stored in these systems may not be accurate due to the data input and collection methods used. For example, a large portion of LAWS data is entered manually and therefore may be inaccurate and incomplete.

B. PURPOSE

The purpose of this research is to ascertain the feasibility of developing a prototype relational database system for storing and managing FBE data. Such a database system would enable analysts to retrieve data and information from past experiments for the purposes of analysis and extracting lessons learned to benefit future experiments.

This research is part of a larger effort to develop a knowledge base system to support FBEs by transferring knowledge obtained from past experiments to the appropriate context of a current experiment. To accomplish this elaborate task, a four-phased approach is being considered.

The first phase, the focus of this thesis, develops a data model and application for storing and managing the quantitative aspect of FBE data. More specifically, this first phase provides:

1. Collection of the data and functional requirements needed for the data model and application system.

2. Development of a data model that consists of the entities, attributes, and relationships of FBE data.
3. Development of an application system that defines the main queries, forms, and reports of the developed data model.
4. Development of data import facilities that populate the developed database with data from other archival systems (e.g., LAWS).
5. A prototype data management system tested with actual FBE data to evaluate its functionality, usability, and effectiveness.

The second phase develops an ethnographic model to collect and manage the qualitative aspects of FBE data. The third phase integrates the first two efforts into a loosely defined "Knowledge Management System." The fourth phase evolves the product of the third phase into a full-fledged knowledge management engine and system using an appropriate technology (e.g., object-oriented, Web, or collaborative technologies).

C. SCOPE AND METHODOLOGY

This thesis focuses on the collection, storage, and analysis of tracking and targeting data. Development of the system involved designing a schema (data model) that consisted of entities, attributes, and relationships of the FBE environment and structure (both physical and logical). The personnel structure was studied for data model completeness, but not implemented in the prototype system.

The data model is transaction- (event-) based and concentrates on information flow in order to categorize and store the data. It represents the information exchanges and data links that are the focus for this research. This approach provides logical links between the identified entities and the capability to query the system for desired information. A prototype application was coupled with the database to complete the data management system and facilitate data entry, modification, and querying.

The following research methodology was implemented to properly develop the

system. The architectures of fleet battle experiments and war games were studied. Elements and entities of the architectures were discussed to establish their significance to the overall structure, and personnel were interviewed to determine information flow and critical points of information exchange.

A schema was developed in an attempt to capture the essence of the FBE architecture. The schema was analyzed to ensure that critical entities were accurately defined and that key information was not overlooked. Relationships were studied to guarantee proper entity interactions and dependencies.

Upon data model approval, some requirements were gathered for application implementation. Application interaction with the database was then tested and requirement completion verified.

This thesis is organized as follows. Chapter 2 discusses the overall framework of FBEs, data sources, and collection methodologies. Chapter 3 discusses the data model developed and its components. Chapter 4 addresses the application created and its capabilities. The final chapter addresses benefits, shortcomings, and lessons learned and provides some recommendations for improvements to the system.

II. OVERVIEW OF FLEET BATTLE EXPERIMENTS

A. FRAMEWORK

The Chief of Naval Operations (CNO) established the Maritime Battle Center (MBC) at the Naval War College (NWC) in Newport, Rhode Island, to serve as the single point of contact for Navy Fleet Battle Experimentation and participation in Joint Experiments. This action was the first step in streamlining and invigorating the Navy's warfare concept development, doctrine refinement, and warfare innovation process.

The MBC is responsible for designing and planning Fleet Battle Experiments, coordinating the execution of these experiments in conjunction with the numbered fleet operational command elements (OCE), and analyzing and disseminating experiment results. The FBE results are used to accelerate the delivery of innovative warfare capabilities to the fleet, identify concept-based requirements, and evaluate new operational capabilities.

The Navy Warfare Development Command (NWDC) was officially established on 10 August 1998 in Newport, Rhode Island. [Ref. 11]

B. DESCRIPTION

Each FBE has a hypothesis and specific, carefully considered measures of effectiveness. FBEs produce, new doctrine, new insight into technology in an operational environment, identification of new required operational capabilities, identification of new acquisition requirements, ideas for further warfare concepts, and ideas for further experimentation.

Unlike a war game, a FBE involves real components and assets. Military units are employed and the exercise is executed in a real-world environment. However, some computer systems may be used to supplement the real environment by 'injecting' synthetic, or virtual, targets into the experiment.

C. ORGANIZATIONAL STRUCTURE

Two types of organizational structures can be examined. One is the authorities and personnel that design, plan, and execute the FBEs and the other is the organizational structure of the FBE itself. The organizational structure of the experiment is adaptable to the scenario and objectives addressed by the experiment's concept of operations. Therefore, the objectives determine the procedures followed by the personnel taking part in the experiment and the flow of information during the exercise.

1. Design, Planning, and Execution

The structure of the authorities ruling over the FBE organization and its procedures begins with strategic level involvement.

The MBC is a CNO sponsored command chartered to conduct FBEs that examine new technologies and new operational concepts in the context of the Navy of the future. The results of FBEs are used to accelerate the delivery of innovative warfare capabilities to the fleet, identify concept-based requirements and evaluate new operational capabilities. [Ref. 10:p. 2] The MBC is responsible for designing and planning FBEs, as well as, coordinating the execution of these experiments in conjunction with the numbered fleet operational command elements (OCE), and analyzing and disseminating experiment results.

2. The Experiments

Although FBE architectures may differ due to changing objectives, the basic structures are similar. This research was based mostly on FBEs E, F, and G, with a concentration on FBE-G.

The MBC and the Commander, U.S. SIXTH Fleet (C6F) jointly conducted Fleet Battle Experiment Golf (FBE-G). C6F Staff and the Navy Warfare Development Command focused on large themes of dynamic sensor management, decentralized

operations, and sensor, actor, and decision-maker synchronization. These themes were chosen to support an investigation of the emerging Network Centric Warfare (NCW) hypotheses in the pursuit of dynamic attack operations. [Ref. 10:p. 68]

NCW focuses on shortening the Detect-to-Engage timeline. A critical part of shortening the timeline is being able to respond effectively against emergent high threat targets, or Time Critical Targets (TCT).

The experiment focused on the Time Critical Targeting process using a dispersed command, sensors and engagement architecture to allow forces to respond quickly to fleeting targets and allow them to commit weapons and move sensors with knowledge of the impact those decisions would have. TCTs are a subset of Time Sensitive Targets. Time Sensitive Targets are fleeting targets that can only be effectively engaged during limited periods of time. TCTs are targets requiring immediate response because they pose (or soon will pose) a clear and present danger to friendly forces or are high-value fleeting targets of opportunity. [Ref. 9:p. 55]

The primary goal was to apply the concepts of NCW to the Joint Task Force (JTF) structure. It included Intelligence, Surveillance, and Reconnaissance (ISR) asset management and allocation of resources, the use of ASW search methodology for land targets, weapon apportionment and sensor-weapon-target pairing. FBE-G applied a distributed Command and Control (C2) architecture, in contrast to the centralized C2 architectures of earlier experiments. [Ref. 9:p. 4]

Figure 1 depicts the traditional C2 structure for FBE-G with the inclusion of the concept of Maritime Control Centers (MCC). It is intended to empower individual units at various levels in the traditional C2 structure to act autonomously in the engagement of TCTs. The decentralized network is used to facilitate the process. While the traditional C2 structure remains in place for deliberate strikes, a network centric approach is employed in the specific case of TCTs to accelerate the speed of effects. [Ref. 9:p. 15]

A Maritime Control Center is a combined sensor, engagement and command node that has been allocated and apportioned three key assets: Battlespace; Resources, both in sensors and weapons; and Autonomy of Action within Commander's Guidance and Rules of Engagement. [Ref. 9:p. 16]

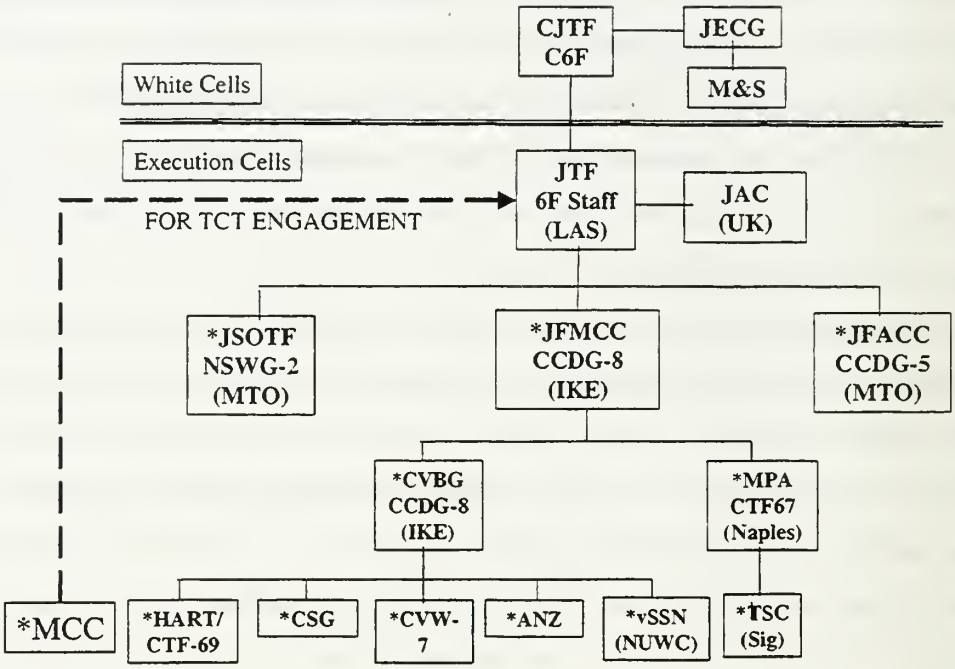


Figure 1 Command and Control Organization for FBE-G with MCC Concept Included (MCC Units Designated with Asterisk). After Ref. [9]

Information was disseminated throughout the forces taking part in the exercise and each unit was granted some autonomy in deciding if they could effectively engage the targets. [Ref. 9:p. 5-7] This method of tasking was used in hopes of improving reaction times and effectiveness in engagements.

Figure 2 is a representation of the sensor network for FBE-G. The sensor grid was composed of decentralized, distributed sensor nodes. Sensor nodes were controlled/focused using distributed collaborative planning and a shared workspace. Sensor nodes directly controlled sensors or had access to the controllers for near real time flexing of assets. Sensor reassignment and geolocation requests were shared through a common workspace; sensor management chat-rooms. [Ref. 9:p. 9]

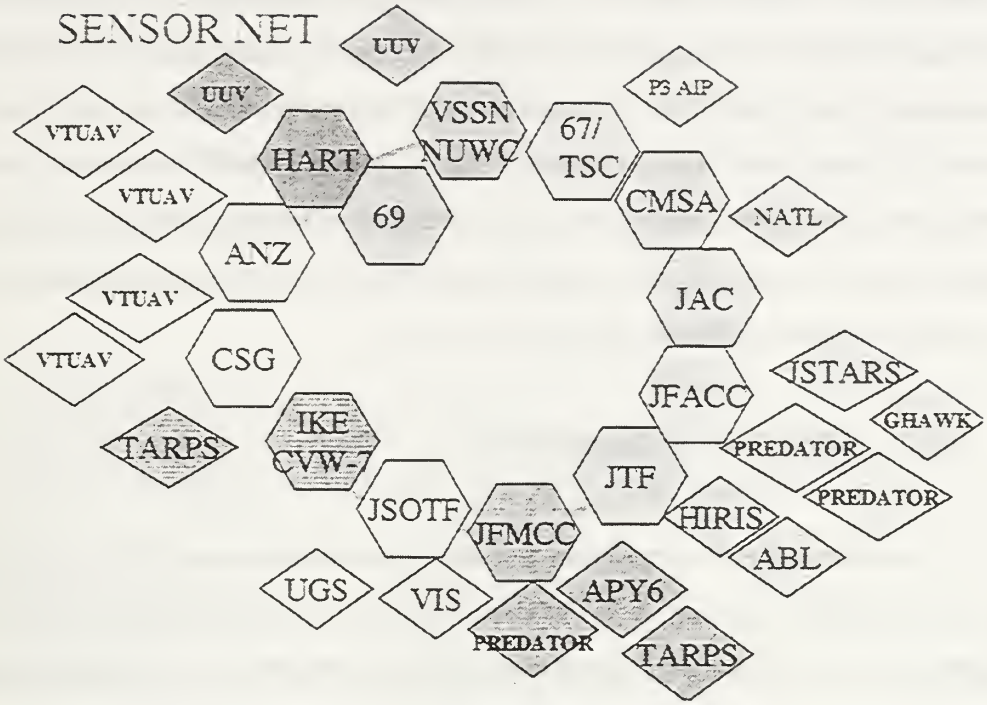


Figure 2 Sensor Network for FBE-G. After Ref. [9]

D. DATA OF A FBE

The amount of data produced during a fleet battle experiment can be overwhelming. Although extremely diverse in its types, formats, and sources, the data extracted from a FBE can be classified under two main categories: quantitative and qualitative data.

Quantitative data makes up the greater part of the data and is the focus of this project. Quantitative data is typically raw, unprocessed data such as times, quantities, and locations. This type of information can be classified, stored, and queried more easily than qualitative data. It is factual, precise, and indisputable. Few things can be done with quantitative data except using it for decision support and assessment substantiation. It includes information on track, target and platform locations, munitions used, times of events, and other detailed information.

Qualitative data includes opinions and assessments developed by personnel after

examination of massive amounts of information (quantitative data). Qualitative data is the result of analysis and is derived by experts based on their examination of the quantitative data. It may be a determination of the relevance or condition of an event or action. This kind of data is not always obtained from a specific source and may not be exact, factual, or precise in nature. It is debatable and irregular in its configuration, making it very hard to store for future referencing. It is filtered and assessed by knowledge experts and data collection personnel.

1. Sources and Collection

Fleet battle experiment data is produced at numerous locations by several sources. Sources of qualitative data include both civilian and military personnel taking part in the experiment as well as data collection and analysis personnel. Assessments, decisions, and other factors can be used to determine effectiveness and correctness of actions in addition to ascertaining if a problem was encountered in a system or process.

Quantitative data is provided by, or can be extracted from, systems such as Land Attack Weapons System (LAWS) and Global Intelligence, Surveillance, and Reconnaissance System (GISRS). LAWS and GISRS provide the capability to manage weapon and sensor nodes real time, respectively. Tables 1 and 2 list the engagement and sensor nodes (platforms) and assets (weapons, sensors) contained by each.

Platform	Weapon
USS Anzio	TLAM/TTLAM/ERGM/LASM
USS Cape St. George	TLAM/TTLAM/ERGM/LASM
USS Hartford	TLAM/TTLAM
NUWC (sim SSN)	TLAM/TTLAM
P-3	SLAM-ER
TACAIR from USS Eisenhower	JDAM/JSOW/PGM/SLAM-ER
TACAIR from JFACC	JDAM/JSOW/ PGM
SOF	Direct Action.

Table 1 Engagement Nodes and Assets for FBE-G.

Systems such as LAWS and GISRS, though they seemingly categorize and sort

data, are still reliant upon personnel for data entry. This introduces the possibility of data input irregularity and inaccuracy. Much information is left blank due to the rapid pace of activity during the experiment. This introduces the problem of data absence, which must be handled by the data collection personnel at a later time. They correct these problems by either inferring actions and results or attempting to gather the information from other sources, then back logging the information into the systems.

Location	Sensors
CJTF (C6F onboard LaSalle)	NTM, U-2
JFACC (CCDG 5 on MTO)	JSTARS GS, Global Hawk, Rivet Joint, HIRIS
JFMCC (CCDG 8 on USS Eisenhower)	APY-6, TARPS, Predator, EP-3
USS Anzio	VTUAV,
USS Cape St. George	VTUAV,
CTF-69/USS Hartford	ELINT, SOF
CTF-67/TSC Sigonella	AIP P-3
JSOTF (cell on MTO)	SOF, UGS
JAC Molesworth UK	All National and Theater Sensors

Table 2 Sensor Nodes and Assets for FBE-G.

Other sources include records kept by personnel during the execution of the exercise and interviews or discussions held afterwards. Information gathered from personnel will tend to be more imprecise and difficult to sort, categorize, and store appropriately due to variation.

The primary concern for the latest experiments has been track acquisition and reaction time reduction for assessments. Track acquisition, assessment, and engagement events produce the majority of the information gathered about the exercise. However, information concerning weapon utilization and effectiveness, message delays, and situational awareness is also determined and gathered by personnel.

Manual recording of information by FBE personnel constitutes the bulk of the information collection effort. Data concerning tracks, targets, platforms, decisions, and many other factors is recorded on either forms that personnel are afforded or is entered into data repository systems such as LAWS.

Because of this methodology, data collection is inconsistent and the data entered may vary based on the person recording. This irregularity in data formats produces another problem at the time of data analysis.

The latest version of LAWS, demonstrated in FBE-G, was utilized with some problems and many records were incomplete and inaccurate. Time formats were not clearly defined and fields were frequently left blank. Because of this, few tests could be run to determine the effectiveness of data analysis concerning the event timeline.

Target Location Error information, provided by mensuration events, was entered into the remarks fields along with other random, unrelated information. Other information, such as platform identification and threat types, was formatted improperly making data filtering very difficult. Data transfer into event entities was also problematic and complicated because of the varying ways the data was recorded.

2. Storage

The classification of the data is an arduous task. Current methods have personnel studying this data and then categorizing it by source, subject, and type.

Data storage is of great concern because access to data from past experiments could be used in designing and performing future experiments as well as providing decision support to participants. Analysis of data from various experiments is nearly impossible because of the sheer magnitude of information. Knowledge management experts and data collection personnel work hand in hand to try to find trends in the data and to determine difficulties in the processes. Difficulties could be attributed to system overloads or to human error. By storing data more effectively, the hopes are to make data access easier and data collection and categorization faster so data comparison and analysis can be performed more effectively and efficiently.

Databases have been created to store the data from FBEs, but with limited success. These databases are designed mainly for storage of text strings. The information is categorized after the experiment is executed. Personnel collect, analyze, and categorize the data (mostly comments and assessments made by FBE participants)

based on the subject matter, the sources, and the conclusions made. This data is labeled accordingly and stored in a database that can reference it at a later date by those categories.

This provides few querying options. The data can only be extracted in the form in which it is entered. Few, if any, conclusions can be drawn from the information and searches are predetermined to fit a specific request because no real links, other than categories, can be made between the different pieces of information.

Data provided by systems, such as LAWS, is stored in those systems to be studied and disseminated by personnel. Associations between data in separate systems can only be made through human involvement. There is no existing way to manipulate and link the data across differing systems. Trends and logical relationships are difficult to determine, if at all possible.

3. Information Flow

The flow of data begins at the time of acquisition. A sensor acquires a track and information pertaining to the track, sensor, and time of acquisition are all logged. Once the target has been positively identified, it can be redesignated from a Unidentified Assumed Enemy (UAE) state to Hostile (HOS). At that point, the HIT will be redesignated as a target (TGT). This status will be updated in the Digital Track Folder (DTF) and made available to all engagement cells. [Ref. 9:p. 19]

As shown in Figure 4, the track is then submitted (nominated) as a possible target by a GISRS terminal. Targets that are being actively prosecuted by a sensor will be additionally labeled as a High Interest Track (HIT). This will indicate that target mensuration and refinement are in progress. [Ref. 9:p. 19]

Mensuration is used because some tracks may require more detail, or more precise information, pertaining to location, altitude, threat type, etc. A request for mensuration may be sent by the acquiring terminal to a Precision Targeting Workstation (PTW) or a Joint Targeting Workstation (JTW). Once a nomination is requested and mensuration information received, the track is determined to be a viable target or not. If

evaluated as a viable target, a fire command with pertinent information is sent to all platforms available. Providing all assets with as much information as possible so that, through autonomy, the most effective solution can be achieved is the net-centric ideology. Geosolution is considered complete when the target solution is good enough to engage with at least one available weapon. [Ref. 9:p. 19]

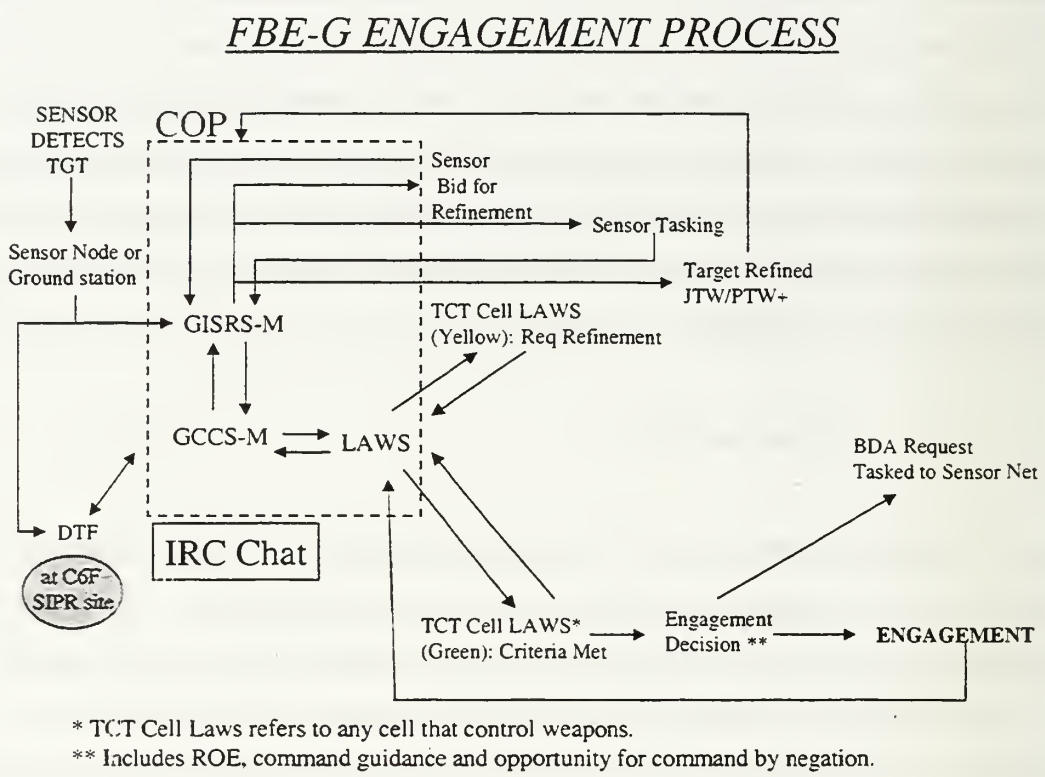


Figure 3 Detailed Engagement Process. After Ref. [9]

The distinction between tracks and targets hinges on the assignment of a weapon to a track. FBE-G operations outline that a track was not considered a target until it was assigned a weapon. A platform to engage a target is assigned upon issuance of a fire command. Figure 5 displays the engagement net used for FBE-G consisting of the platforms and the weapons contained within them. For FBE-G all platforms were issued a fire command upon target acknowledgment. The "Target-Weapon" pairing, as referred to by FBE personnel, is represented in the schema by including the Weapon Type

attribute within the Target entity.

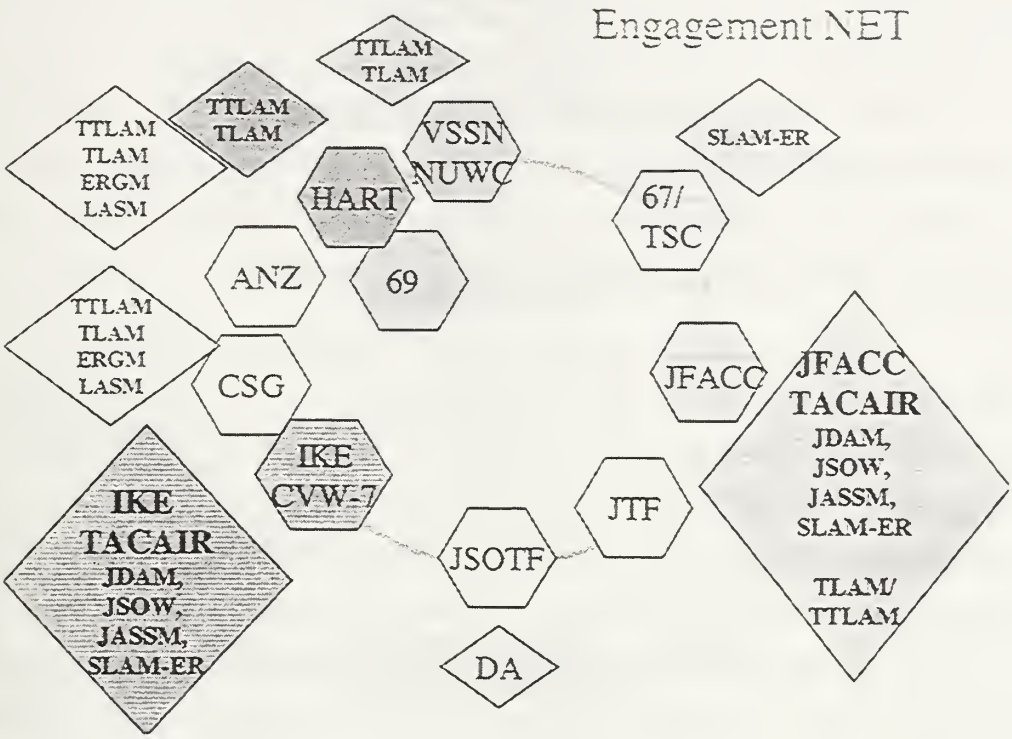


Figure 4 Engagement Network. After Ref. [9]

At this point the autonomy aspect of the experiment is apparent. Once provided with firing information and given the authorization to fire, the individual platforms may determine if they are to engage the target. Upon engagement, or firing, information is continuously exchanged between platforms, sensors, and other systems to determine impact and assessment of the target after impact.

All this information is time-stamped in order to be able to reproduce a time line of events at a later date and to determine any information bottlenecks, problems, or variation. This information is also useful in analyzing decision-making skills by the commanders and other decision-makers involved in the experiment.

The information flow is reproduced in the database schema through the use of key fields that relate each entity and/or event. Time attributes in each object provide the timeline information that is crucial to event recreation. Figure 5 provides an overview of the TCT process from track acquisition to target engagement.

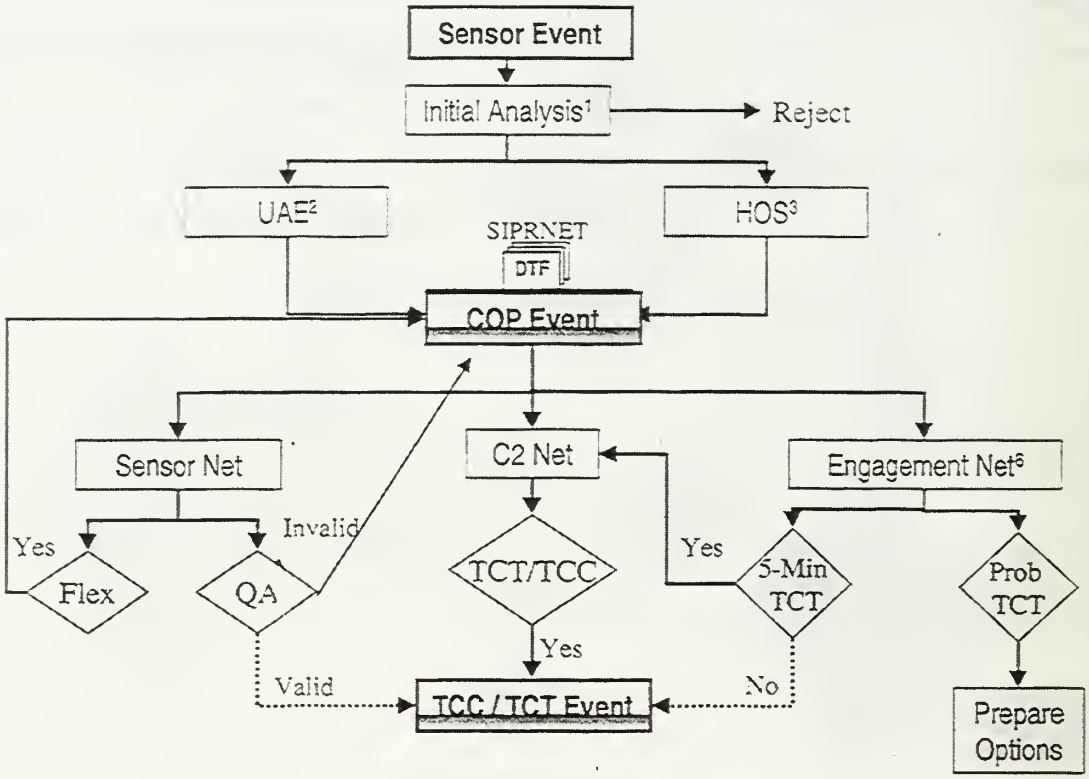


Figure 5 TCT Process Overview. After Ref. [9]

III. DATA AND APPLICATION DESIGN

Databases have been utilized to store the data from FBEs, but with limited success. These databases are primarily used for storage of text, and searches on these databases are subject-based, not event-based. No practical attempts had been made in designing a schema that would effectively support storing and querying FBE data and provide customized results to meet user needs.

Relational databases store data in tables and enable applications written against this data to retrieve and update data from these tables. In order to maintain an effective and efficient database, data redundancy and inconsistency is eliminated by ensuring that fields related to the same category of data are stored in the same table. Relationships between tables allow for retrieval of related data stored in different tables. The types of relationships supported by a relational database are One-to-One, One-to-Many, and Many-to-Many. Many-to-Many relationships are implemented by creating two One-to-Many relationships and using a third table as an intersection to store the relationship between the two tables. Some common examples for each relationship are Employee-Job, Department-Employee, and Students-Classes, respectively.

Normalization is an important concept to relational databases that eliminates inconsistencies and minimizes inefficiency. A fully normalized database stores each piece of information in the database once with each entity represented in a table that is uniquely identified by its own primary key. A primary key is an attribute that uniquely identifies a record. Normalized tables allow users to reference any piece of information in other tables by linking them together through the foreign keys. A foreign key is an attribute in a table that links to the primary key in another table.

A. SCHEMA DESIGN

A schema is a mapping of the database. It diagrams all the tables, fields, and relationships in the database. The schema for the prototype database was designed to

account for all entities and relationships that represent the architecture of the FBE. (See Figure 6.) This includes personnel, command cells, and roles occupied by personnel. Although these entities are not the primary points of interest of the prototype system, they are included in order to provide a complete picture of the structure and to allow for growth of the system in future development stages.

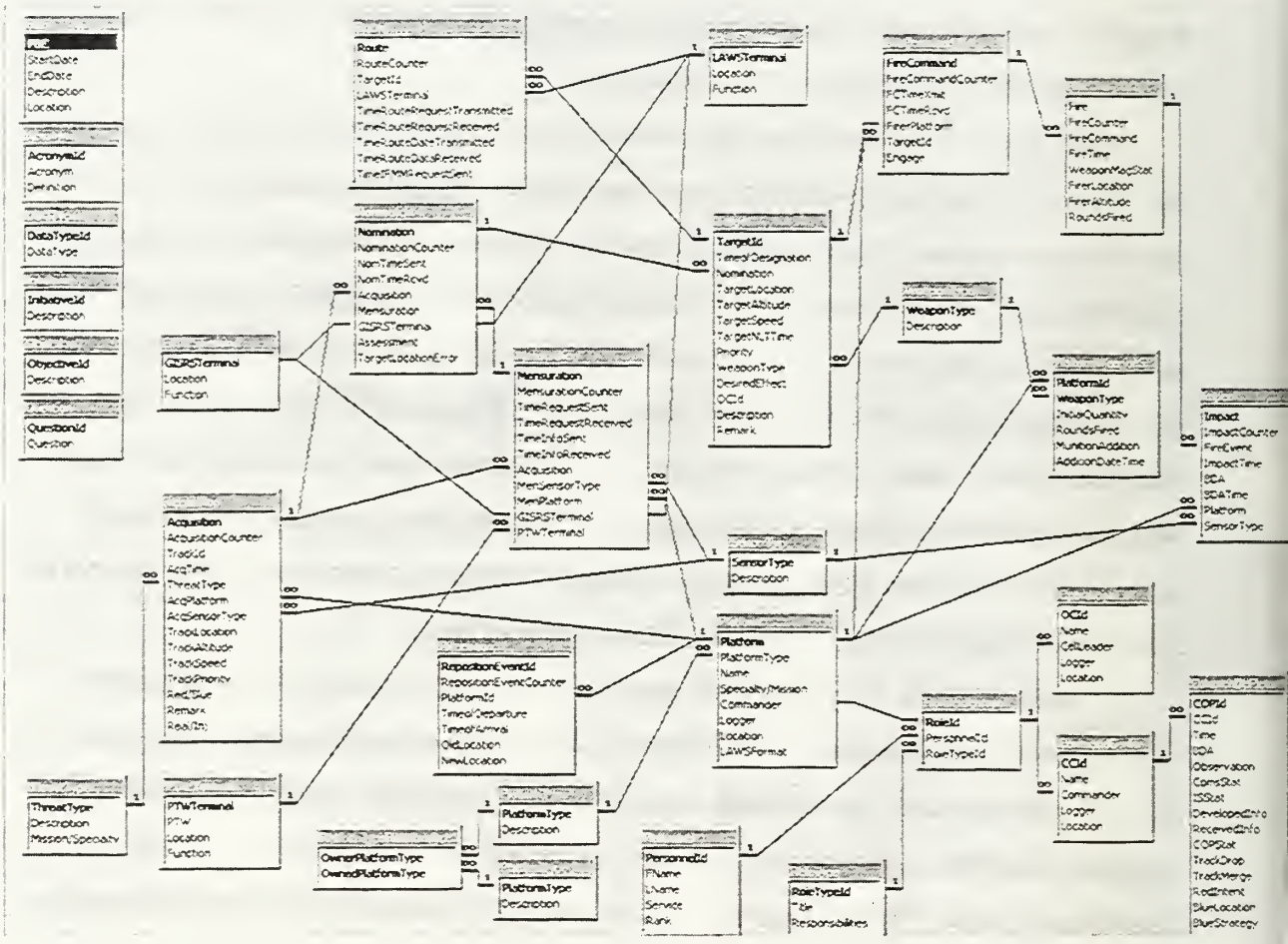


Figure 6 Fleet Battle Experiment Database Schema with All Entities Represented.

As previously mentioned, the schema is transaction- (event-) based. The data model represents the interactions between different systems and information sources by concentrating on the logical events and the information transferred rather than the physical objects manipulating the data.

Entities represented in the data model represent both logical and physical objects. Event entities such as track acquisitions, target nominations, firing commands, and target engagements are events that occur during the experiment and are examples of logical entities in the data model. Physical entities are those such as platforms, sensors, weapons, and personnel.

In the remainder of this section we describe the main entities of a FBE and how they relate to each other. These core entities include Platform Type, Sensor Type, Threat Type, Weapon Type, weapon, sensor, and mensuration information managers, Acquisition, Mensuration, Nomination, Target, Route, Fire Command, Fire, Impact, and Reposition events, operational cell and command center nodes, and personnel.

- The **Platform Type** entity defines the general types of ship, aircraft, or other vessels that will participate in the FBE. Information related to a type of platform is stored in this entity. For example, the USS Dwight D. Eisenhower is a specific vessel of type CVN. A platform is involved in acquisition, mensuration, fire command, impact, and reposition events. Since a platform type can contain other platform types as assets (i.e., a cruiser owns a helicopter as an asset), a recursive relationship needed to be established for the Platform Types entity.
- A platform type entity can be related to many **Platform** entities which describe specific platforms.
- The **Threat Type** entity represents the various expected threats that will play a part in the FBE. A type of threat can be part of many instances of acquisition events and is therefore a part of the acquisition event definition.
- The **Sensor Type** entity defines the generic sensor types that will be used in the FBE. Any type of sensor can be used for multiple acquisition, mensuration, and impact events.
- Similar to the Sensor Type entity, the **Weapon Type** entity describes a generic weapon that will be used in the FBE. A weapon type can be linked to many platforms and can be used against many targets.
- A **Mensuration** event is the request for more precise information on a

particular track. An acquisition event may produce more than one mensuration event but a mensuration event is requested by a specific acquisition event and a GISRS or LAWS terminal. Mensuration requires the involvement of a platform and sensor, which may or may not be the same as the acquiring platform and/or sensor, and a PTW terminal. Any platform, sensor, and PTW terminal may be involved in more than one mensuration request. The inverse is not true, however.

- Weapon, sensor, and mensuration information managers are known as **LAWS**, **GISRS**, and **PTW Terminals**, respectively. These terminals are used during mensuration, nomination, and route events.
- An **Acquisition** event defines the acquisition of some threat or the creation of a track by a sensor. This entity is an intersection of the threat type, sensor type, and platform entities. It links a threat type with its acquiring sensor and the platform on which that sensor is located through the use of foreign keys from those entities. A platform, sensor type, and threat type can be linked to many acquisition events, but a specific acquisition event may only posses one platform, sensor type and threat type.
- A **Nomination** event is the recommendation from some GISRS or LAWS terminal to designate an acquired track as a target. As with the mensuration event, it is initiated by an acquisition event. The nomination requires information from the mensuration event in order to arrive at a valid assessment of the track and determine if it should be considered a target. A nomination also requires a GISRS terminal for the data analysis.
- A **Target** is a track that has been assigned a weapon by a nomination event (target-weapon pairing) and designated as a target for engagement. A target therefore links the track from an acquisition event (by way of a nomination event) and a weapon type. A specific nomination event can produce numerous targets if multiple attacks on the track are desired. A weapon type can be used in many target-weapon pairings and therefore linked to many targets. If more than one weapon is assigned to a track then the different

assignments are appointed different target ids.

- **Route** events are the product of a LAWS terminal and contain flight and target information specific to the target-weapon paring of a Tactical Tomahawk Land Attack Missile to a target.
- The **Fire Command** entity is an event identifying the assignment of a target to a specific platform (not a platform type). A target could create many fire commands (be assigned to many platforms), but the data used in this research only provided examples of one firing command per target id.
- A **Fire** event is the actual firing of a weapon by a platform, or the engagement of the target. A firing command usually produces only one fire event because a firing of multiple rounds is recorded as one firing and the number of rounds is recorded as information for that event.
- An **Impact** event is the result of a firing event. The assessment of the impact is part of this event and involves the use of a platform and a sensor for gathering data.
- In order to keep information concerning weapon magazine information on platforms, an entity called **Platform-Weapon Status** was created. This links a specific platform with the weapon types used and could help track weapon usage during FBEs.
- The **Reposition** event is used for information concerning platform movement during an FBE. Its information is specific to one platform for a specific range of time and can be used to track asset management and platform availability after the conclusion of the FBE.
- **Command Centers** represent authority units in the FBE that make critical decisions and assessments. They control strategic decisions and asset management. The Common Operating Picture (COP) is in reality an assessment made by personnel who have studied information about the environment. It is the product situational awareness. The COP is defined by Command Centers.

- **Operational Cells** are similar to Command Centers but tend to control specific actions and platforms. They provide lower-level decision making power. They are linked to targets because of their influence in determining the viability of a target and the assignment of platforms to targets.
- The **Personnel** entity contains information about specific individuals such as name, rank, and service. These individuals may be assigned many jobs, or **Roles**. Many Roles have the same titles because they perform the same tasks and responsibilities but on different platforms. Therefore, personnel and their roles are linked through the use of an entity called **Role Assignment**. This entity can then be related to specific platforms and cells as necessary.
- The **FBE** entity was created to store FBE specific information such as dates and location for the experiment. The entities **Acronyms**, **Data Types**, **Initiatives**, **Objectives**, and **Questions** hold respective information pertaining to the experiment. They are entities created to store specific information and have no necessary relationships with any of the other entities.

B. APPLICATION DESIGN

Due to the lack of user defined requirements for application implementation, some assumptions had to be made regarding querying and reporting capabilities.

1. Querying

Due to the novelty of the concept for this system, FBE personnel had a difficult time formulating application requirements and provided few suggestions concerning system-querying capabilities. Several querying options were formulated and presented to the user. These suggestions were acknowledged as likely and beneficial options so they were implemented as predefined, or "canned", queries. In general, these queries extract information on the main entities based on different arguments or parameters. A detailed

description of these queries is presented in the following chapter.

In order to provide some flexibility for users a custom SQL-based query option was also included in the design of the system.

2. Reporting

The same situation occurred with report specifications. It was determined that reports should be provided for query results. Due to lack of requirements, reports that the author felt would be most useful were defined and discussed with the user. A decision was made to make their design as simple as possible in order to allow for future modifications. Simplicity was also a benefit in reflecting the information provided on-screen by the application.

In the next chapter we discuss the implementation of the database and application designed in this chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DATA AND APPLICATION IMPLEMENTATION

A prototype Database Management System (DBMS) was implemented from the data and application design described in the previous chapter. A coupling of Microsoft® Visual Basic and Microsoft® Access 97 was chosen in order to combine the power of Microsoft® Access 97 database processing capabilities with the ease of user interface development of Microsoft® Visual Basic in order to meet the users' needs. The simplicity of connecting the two applications was also an important consideration.

By keeping the database separate from the application, the database could be migrated to other SQL-based databases without the need to re-write or modify the application.

A. DATABASE IMPLEMENTATION

The database for the prototype system was implemented in Microsoft Access 97. It is a relational database management system and was chosen primarily because of its popularity, flexibility, and wide use. Data produced during the FBE is categorized and stored by the logical event that produced it.

Since the storing of data from multiple FBEs in a single database could eventually tax the speed and querying capabilities of the system, it was decided that each FBE would have its own database. In other words, a database will contain data from only one FBE. Some reasons for this decision are 1) it keeps the volume of data in a database to a manageable size so that queries are not slowed and 2) it allows for changes to be made easily for each experiment. The latter reason addresses the differences in FBE objectives and the changing information requirements to support those objectives. A third reason is somewhat related to the latter. If a specific record of an entity is altered between FBEs, then that record's information pertaining to the previous FBE will be lost. This makes the linking of that entity's information to prior FBEs impossible.

The design and military structure of FBEs assigns discrete names or designations

to the units involved, which provides a unique identifier, or primary key, for each entity. This simplified the task of determining entity classifications and ensuring distinctions among instances of the same entities.

Twenty-five tables, representing logical events and physical entities in the FBE architecture, were created in order to accommodate the normalized data. However, other FBE information pieces specific to the FBE design were necessary so additional tables were created. These include FBE attributes (such as dates and title), acronyms, initiatives, objectives, and questions. Two more tables were added to support data import and filtering from LAWS and other possible sources.

For the implementation of the database schema, twenty-one entities were concentrated on with fifteen of those having relationships with each other. I will now discuss the implementation of the entities as tables in an Access database. (Figure 7)

- The **Platform Type** table contains information related to a type of vessel that is used in the FBE. The primary key for this table is the asset's generic acronym used by the military. For example, CVN (aircraft carrier, nuclear), CG (guided-missile cruiser), and SSN (fast-attack submarine, nuclear).
- The **Sensor Type** table contains the generic sensor types that will be used in the FBE. The primary key for this table is the typical name or acronym used by personnel. Examples include UAV (unmanned aerial vehicle) and RECO (reconnaissance personnel).
- The **Threat Type** table contains the various expected threats that will play a part in the FBE. The primary key used is the generic designator used to address a threat such as NODONG (a type of ballistic missile), SAM1 (a surface-to-air missile), or MOBRDR (acronym used for mobile radar). Attributes for threat description and capabilities more clearly define this table.

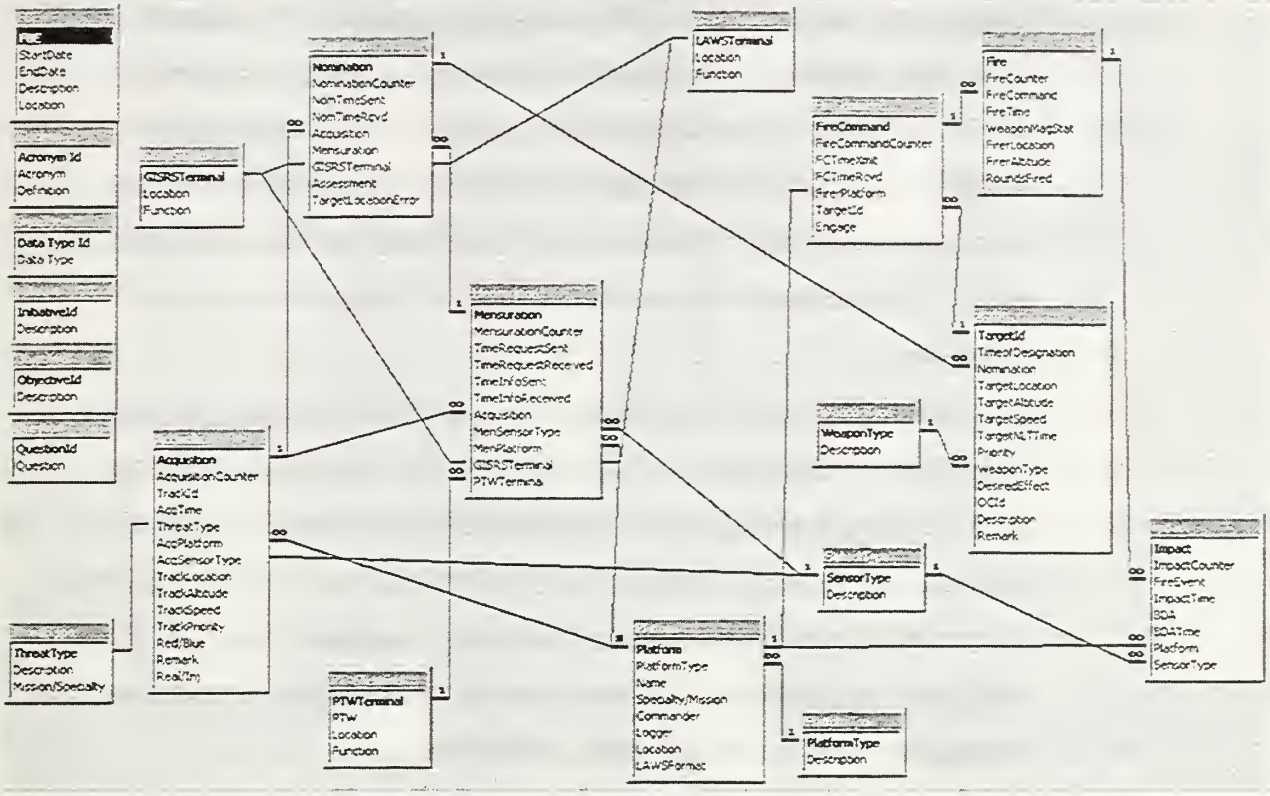


Figure 7 Schema of Entities Implemented in Prototype Database.

- As with the previous tables, the **Weapon Type** table defines a nonspecific entity and contains weapons that will be used in the FBE. The primary key is the military term applied to the weapon. Examples include: AGM (air-to-ground missile), AIM (air interceptor missile), and SLAM (Stand-off Land Attack Missile). Weapon Type also includes a Description attribute.
- **LAWS**, **GISRS**, and **PTW Terminal** tables represent weapon, sensor, and mensuration information managers, respectively. These terminals are used during mensuration, nomination, and route events. The LAWS, GISRS, and PTW Terminal entities all possess like attributes. They are defined by their location and functionality attributes. For future schema development efforts, these terminal types could be implemented as one table with a defining attribute determining the type of terminal as a LAWS, GISRS, or PTW/JTW.
- The **Platform** table contains occurrences of a specific platform type. The

primary key for this table is the platform's designator. For example, the USS Dwight D. Eisenhower is a specific vessel of type whose designator is CVN-69. A foreign key from the platform type table provides information specific to the type of platform. The platforms specific mission or specialty can also be stored in this table. For data importation from LAWS, a table containing the LAWS designation for the platform is also included in the table's definition.

- An **Acquisition** event table defines the acquisition of a track by a sensor. This table's primary key is a concatenation of an abbreviation distinguishing it as an acquisition event and an AutoNumber counter kept by the database. The table contains three foreign keys that links a threat type with its acquiring sensor and the platform on which that sensor is located. Other significant attributes include the track id determined by the acquiring terminal and track information such as altitude, speed, and location.
- A **Mensuration** event table is the request for more precise information on a particular track. The instance of a mensuration event links a platform and sensor, which may or may not be the same as the acquiring platform and/or sensor, and a PTW terminal. Its primary key is also a concatenation of an abbreviation and an AutoNumber counter. Foreign keys from the mentioned entities help define the event. A foreign key specifying a GISRS terminal as an information source is also an attribute of this table. Attributes to store data concerning information request and receipt times are present in this table as well.
- The **Nomination** event table uses a primary key composed in the same manner as the acquisition and mensuration primary keys. Because a nomination requires information from a mensuration event, a foreign key from the mensuration table is included. A foreign key from the acquisition table is also an attribute. An assesment field is provided for qualitative information that may have had a specific influence on the nomination event.
- The **Target** table links the track from an acquisition event (by way of a

nomination event) and a weapon type by including foreign keys from both the nomination table and the weapon type table. The primary key is specified by GISRS and LAWS terminals. Post-mensuration information including time of designation as a target, location, speed, and "No Later Than" time are stored in this table. The "No Later Than" time specifies a time at which, if the target has not been engaged, it ceases to be a time critical target.

- The **Fire Command** table primary key is created in the same manner as the other event primary keys. This table contains the platform and target table primary keys as foreign keys. It also contains attributes stating the time the command was sent and received, as well as if a weapon was actually launched at the target.
- A **Fire** event table is the actual firing of a weapon by a platform, or the engagement of the target. Its primary key is a concatenation of an abbreviation and an AutoNumber. The Fire Command that produced the Fire event is stored in this table using a foreign key. The number of rounds and time of firing is also recorded.
- An **Impact** event table is the result of a firing event. Its primary key resembles those of the other event tables. The assessment of the impact involves the use of a platform and a sensor for gathering data. Foreign keys for both are stored in this table as is the target impact time and the fire event that led to the impact event.
- The **FBE** table uses the FBE name (i.e., E, F, G) as its primary key. It contains the start and end dates and the location of the experiment. The Acronym, Data Types, Initiatives, Objectives, and Questions tables contain memo data type fields that hold respective information that can be search as strings.

Referential integrity requires that records in tables providing information for other tables must exist prior to the creation of the record in the dependent table. Information flow dictates which entities need to be created prior to others. The schema follows the

logical chain of events of the experiment therefore events must be created in the order dictated by the information flow – acquisition, mensuration, nomination, target, fire command, fire, impact.

Data types and formats for all attributes were determined by the information sources and their current formats and usage of the data. Because of the variance in data types and formats table attributes were weakly defined. Most attributes are of text data type to allow for manipulation of the data and a broader variety of formats. This is true of time information. Access and Visual Basic do not recognize time information stored in Military Date-Time format. Therefore, the times in the database are stored as text fields. This means that any analysis or comparison on this type of information must be performed by the application by using string manipulation.

B. APPLICATION IMPLEMENTATION

The application interface for the DBMS was designed and impemented using Microsoft® Visual Basic 6.0: Professional Edition.

The application involved the creation of twenty-eight graphic user interface (GUI) forms that allow the user to add, delete, modify and query the data stored in the database. The application uses both Visual Basic data objects and virtual recordsets in order to access the database. Controls such as text boxes, dropdown list boxes, command buttons, and menus are used to facilitate data entry and modification, as well as form and record navigation.

1. Forms

Upon execution of the application executable program file, a module containing the Main sub procedure is used to determine the database file to be opened by the application. An Open File dialog box, as in Figure 8, is displayed where the user can browse the computer files available for the database file they desire. When the user

selects the database file, the filename is stored as the database name and recordsets are updated by the application to contain the information from the database selected. (Should the user not choose a database file, the main form will be displayed but no search or editing options will be available because there is no database selected. The user can still choose to open a database y using the File menu option.)

The main interface form (Figure 9) is then displayed. The user can now use the application to modify and search the selected database.

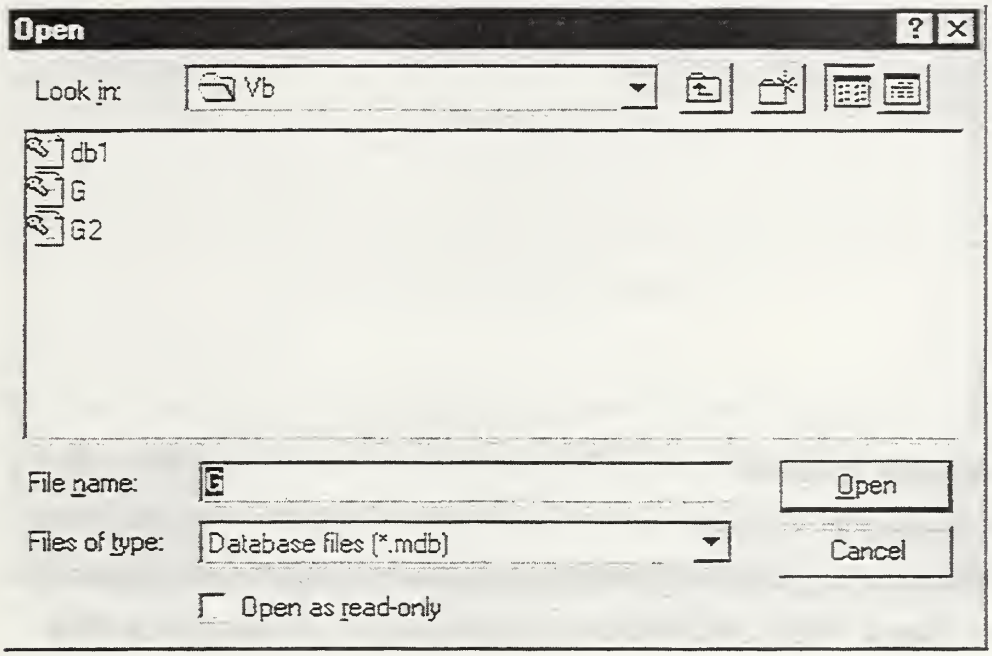


Figure 8 Open Database Dialog Box Displayed at Run-time.

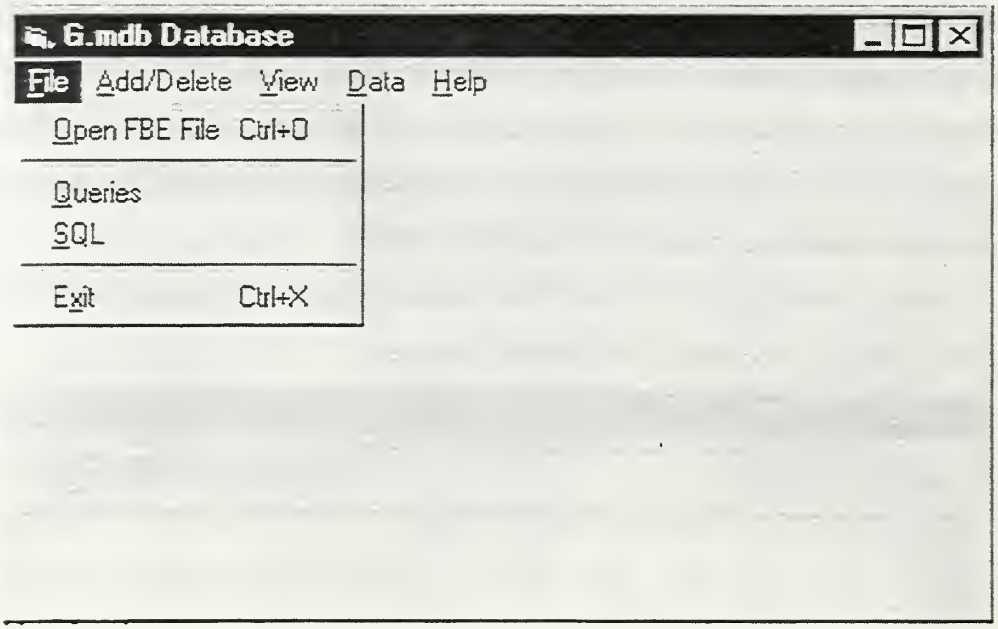


Figure 9 Main Interface Form.

Forms designed for record addition, modification, and deletion (Figure 10) are available to the user under the Add/Delete menu option on the main form. Forms created include Acquisition, Mensuration, Nomination, Fire Command. Fire, and Impact Event Update forms, a Target Record Update form, a Platform Information Update form, Platform, Sensor, Threat, and Weapon Types forms, and Acronym, Data Types, Initiatives, Objectives, and Questions forms.

These forms make use of text boxes and dropdown list boxes to ease data entry and modification as well as displaying update, save, add and delete buttons to manipulate the records. Data objects, which also serve for record navigation, and virtual recordsets are all updated at the time of the form's load event procedure. They provide the data links between the Visual Basic form and the database.

The Add Event button will enable all the data entry fields and clear their contents in preparation for addition of a new record. Upon data entry completion the user must then select the Save Event button to add the record. The Delete Event button will warn the user before submitting the record deletion to the database.

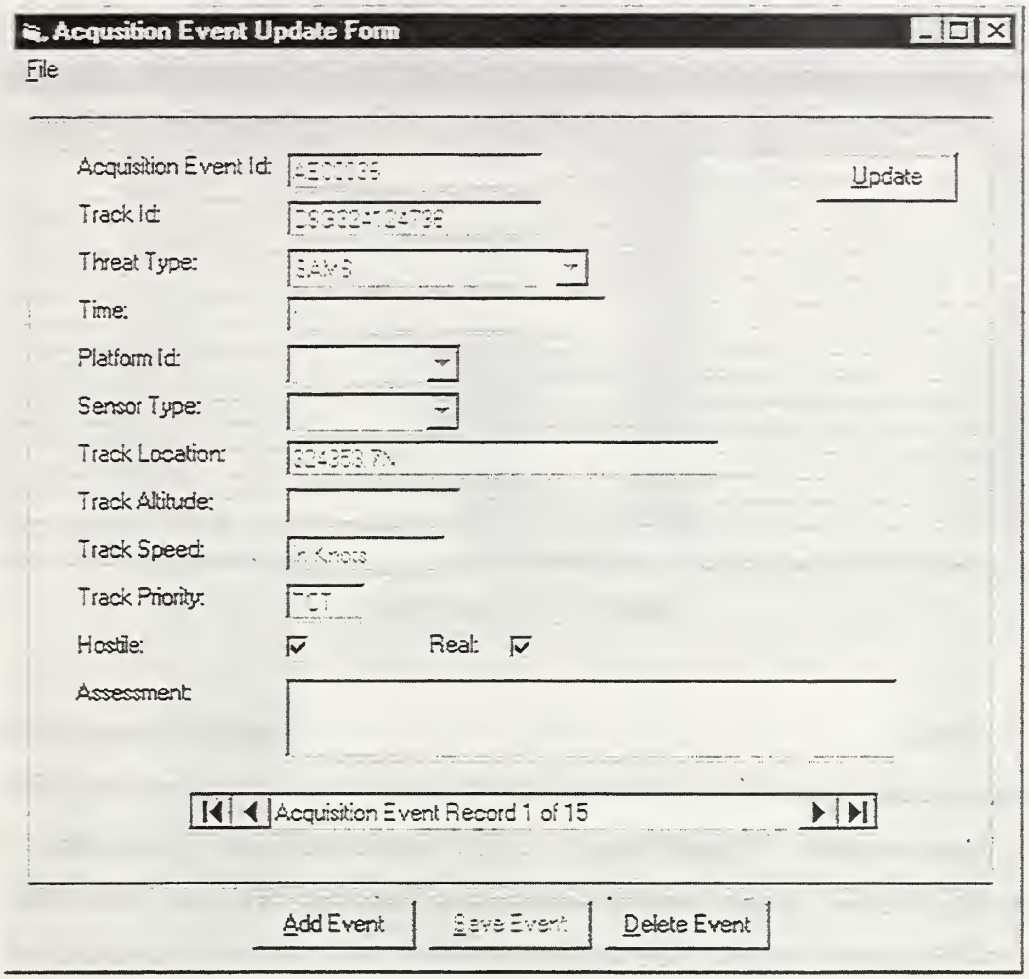


Figure 10 Example Update Form (Acquisition Event Update Form).

To Update or Edit a previously existing record, the user must select the Update button. The Update button's caption will then automatically change to "Submit". After completing all edits, the user will select the Submit button in order to commit the changes.

The user may choose to view all targets stored in the database by selecting the View menu option and choosing Targets. This will display the Targets window seen in Figure 11. When the target window appears the user may display the time line of events corresponding to a specific target by double-clicking the target record they desire. The double-click action will display another window (Figure 12) with the time data for every event pertaining to that target from acquisition to impact.

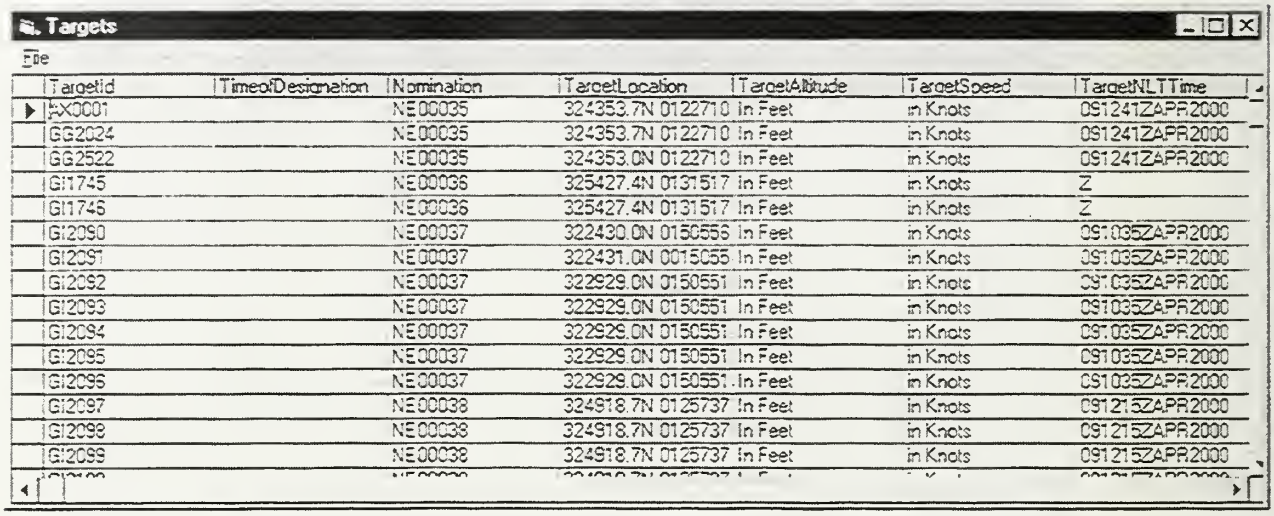


Figure 11 Targets Window.

If the user desires to reduce the number of targets displayed by sorting the records according to some specific criteria, they can choose the filter option under the File menu in the Targets window. This will open the Filters window (Figure 13) and allow them to choose which filters to apply and what criteria to filter with. They can select which filters to activate by checking the appropriate boxes. Once the box is checked, they can then enter or select the criteria they wish to filter with.

The Apply button will activate the filters that are checked but will keep the Filters Window open. The OK button will apply the filters and return the user to the Target Records Window. The Cancel window will return the user to the Target Records Window without altering any filter settings.

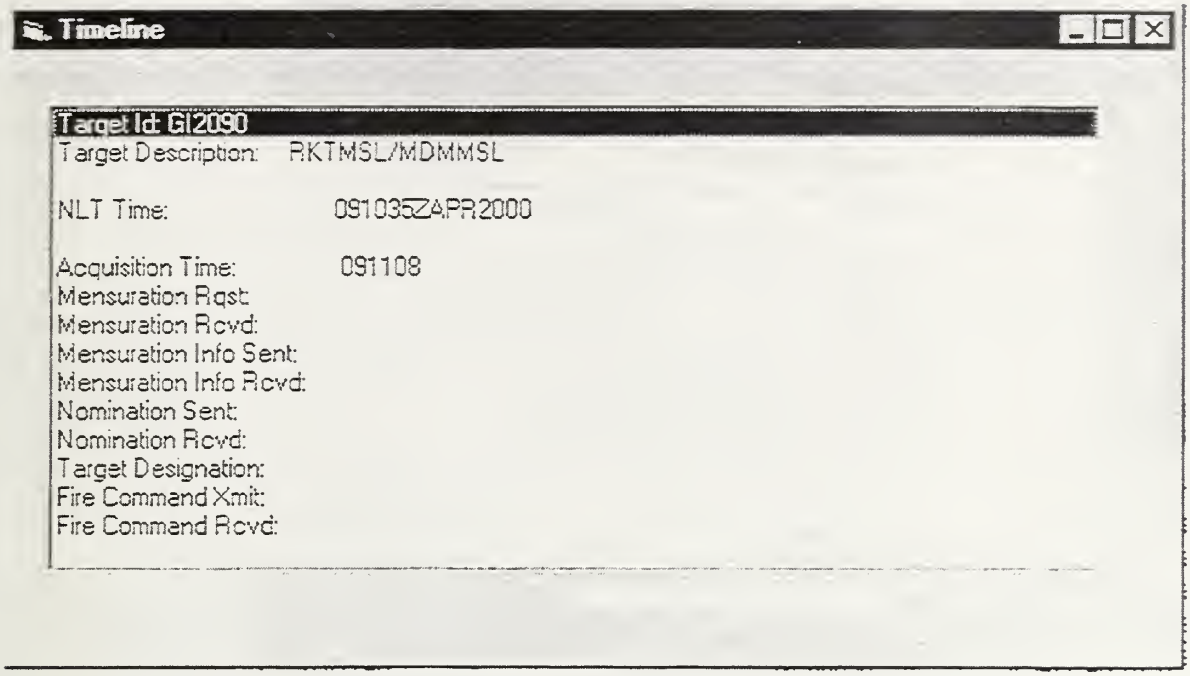


Figure 12 Target Timeline Window.

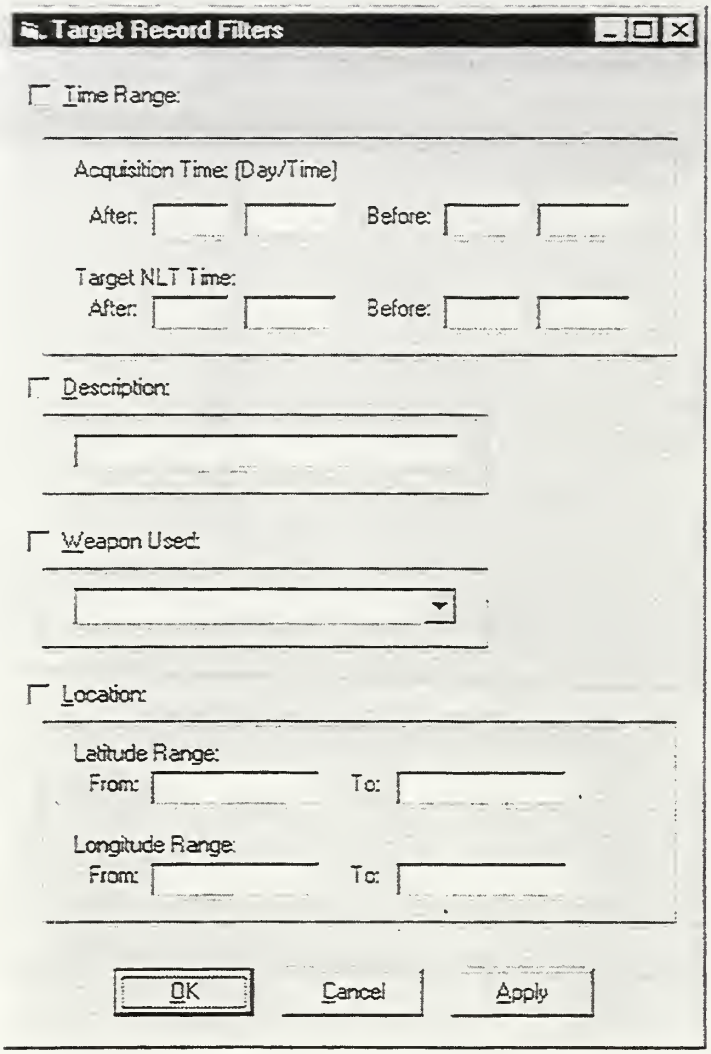


Figure 13 Target Record Filters Window.

2. Menus

The application was implemented as a menu driven application with the main form as the base of form connectivity. By using the main form's menu options users can reach the predefined query form, the SQL custom query form, and all entity update forms. They can also view the list of targets found in the database and even have limited control of the data importation process. They can do all these things with the use of the menus available on the main form. Menu options include opening a different database,

searching the database using "canned" (previously defined) queries or custom SQL queries, editing the database records, and/or populating the database with new information from other systems using text file importation.

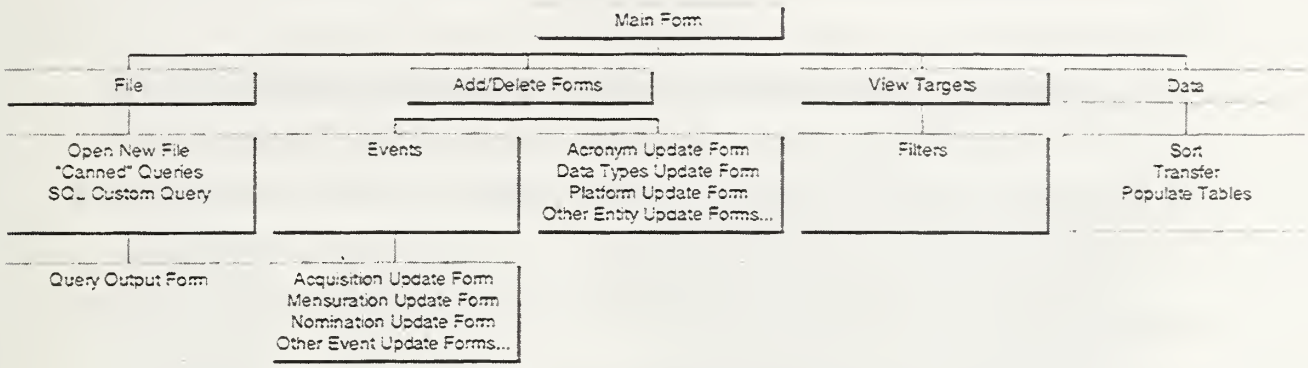


Figure 14 Hierarchical Chart of Menu Options.

The previously mentioned data import capability is based on a capability offered by the latest version of LAWS. The export facility takes user-selected fields from the LAWS system database and exports them to a comma delimited text file. Upon creation of the text file, the information can be imported to a database table. The table used for data import was specifically designed for the exported LAWS data fields, therefore adjustments to the source code would have to be made in order to allow for extra fields or for the integration other systems as data sources. However, the system is operational with the LAWS export text file format. (The database table is named LAWS in the prototype database schema.) Once the table is filled, the data can be sorted and filtered for completeness and formatting using the Transfer and Sort options available under the Data menu item on the Main form. After filtering, the information can be separated and disseminated throughout the database by using the Populate Tables option under the Data menu item. This feature ensures that all the information is sent to the appropriate tables in the database and in the appropriate order as dictated by referential integrity.

3. Queries

If the user chooses the query option from the main form, then another window (Figure 15) displaying the provided queries appears. Option buttons allow the user to select the desired query. When an option button is selected, the appropriate controls will become enabled. The user selects the criteria and runs the query using a command button. The Submit Query button's code module contains a Select Case clause with predefined SQL statements. The option buttons are actually an array of controls whose index determines the case to be selected in the source code and therefore the SQL statement to be used. The form's controls provide the arguments used in the statements.

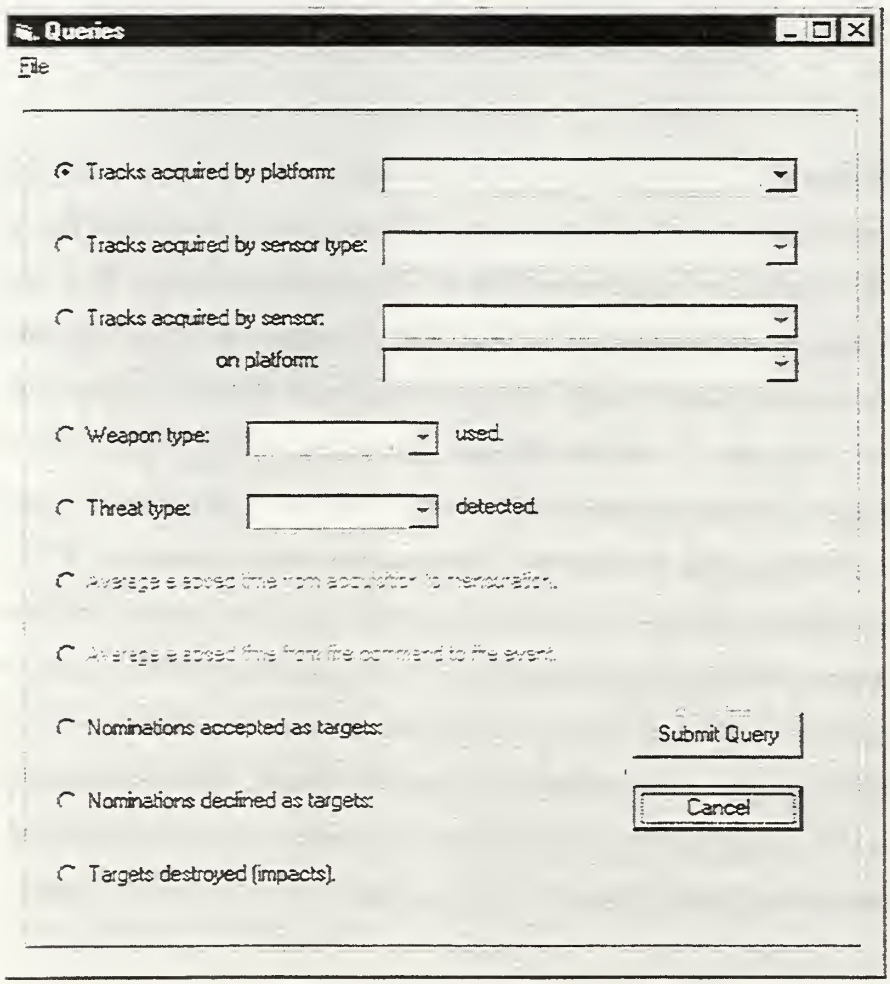


Figure 15 "Canned" Queries Form.

Query results appear in the window shown in Figure 16. Fields displayed are predetermined and set in the source code (hard-coded). These queries do not provide the user with the option of selecting what information is displayed; only the criteria by which to search the database.

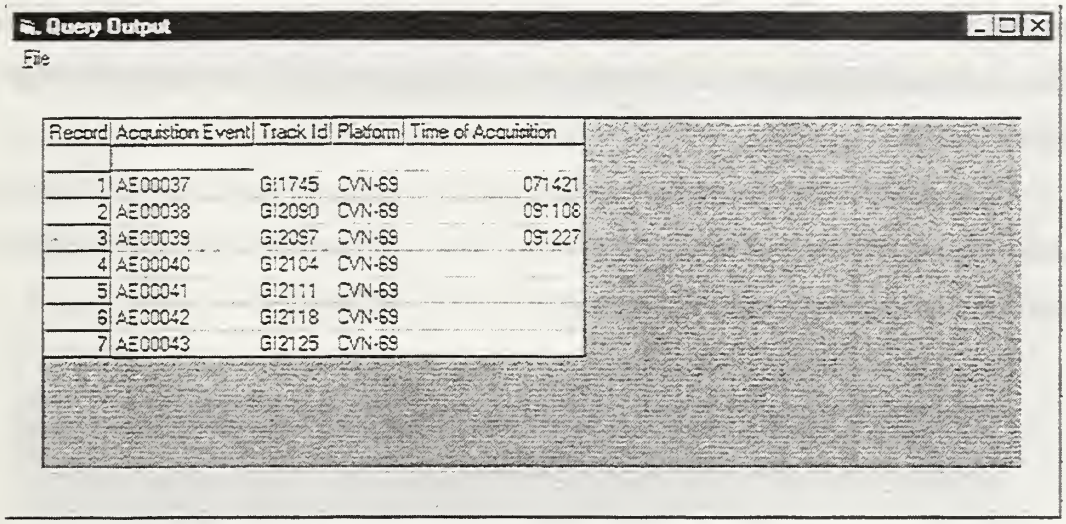


Figure 16 "Canned" Query Output Window.

The user selects which query they would like to use by selecting the appropriate option button. The applicable controls are then enabled and the user chooses the criteria. They then select the Submit Query button and the Query Output Window is displayed with the results of the query.

Four of the eight "canned" queries provided filter acquisition events. These queries filter the acquisitions by the acquiring platform, sensor type, both the acquiring sensor type and platform, and by the threat type of the acquired track. One query filters the target records by the weapon type assigned to, or paired with, the target. Two of queries concentrate on the nomination events – one searching for the nominations accepted as targets and the other searching for the nominations declined as targets. The last query on the "Canned" Queries form retrieves all impact events from the database.

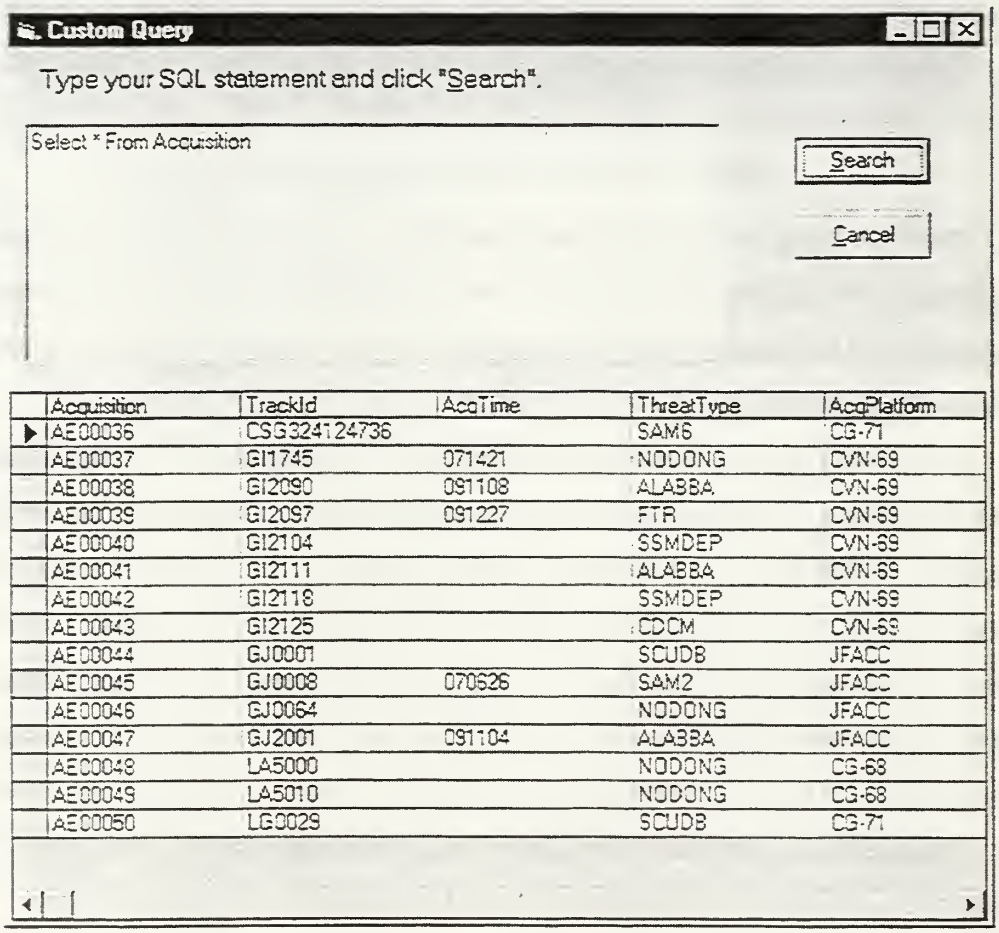
The queries are defined in the source code of the application using cases that contain SQL statements set specifically for the individual querying cases.

For the custom query option, some considerations had to be taken. The biggest

difficulty would have been the presentation of the information, but using Visual Basics Data Bound Grid Control, the fields are updated through the use of a data object that recognizes the SQL-requested fields.

Should the user select the SQL option from the Main form, the SQL Custom Query Window will be displayed. (Figure 17) This window contains a text box in which the user will enter the SQL statement desired. Once the statement is entered, the user selects the Search button and the information is displayed in the Data Bound Grid at the bottom of the form.

Using this query method the user has more control over the fields to be displayed and can alter the criteria to suit their particular needs. However, those who do not fully understand the database schema may encounter some difficulties.



Acquisition	TrackId	AcqTime	ThreatType	AcqPlatform
AE00036	CSG324124736		SAM6	CG-71
AE00037	GI1745	071421	NODONG	CVN-69
AE00038	GI2090	091108	ALABBA	CVN-69
AE00039	GI2097	091227	FTR	CVN-69
AE00040	GI2104		SSMDEP	CVN-69
AE00041	GI2111		ALABBA	CVN-69
AE00042	GI2118		SSMDEP	CVN-69
AE00043	GI2125		CDCM	CVN-69
AE00044	GJ0001		SCUDB	JFACC
AE00045	GJ0008	070526	SAM2	JFACC
AE00046	GJ0064		NODONG	JFACC
AE00047	GJ2001	091104	ALABBA	JFACC
AE00048	LA5000		NODONG	CG-68
AE00049	LA5010		NODONG	CG-68
AE00050	LG0029		SCUDB	CG-71

Figure 17 SQL Custom Query Window.

When querying the database, the primary and foreign keys of each table provide links between the tables. Because the database is relational, table joins must be created to access information from multiple tables.

In order to extract information from a table that is two relationship links away, it is necessary to include the necessary linking information from the intermediary table. For example, suppose you want to query information of an impact event and want to determine either the firing platform or the target fired upon. The links must first relate the Fire event that created the Impact event through the Fire attribute, or field, within the Impact event. Then, to retrieve the platform or target ids that reside in the Fire Command event another link must be made between the Fire event and the Fire Command event by using the Fire Command attribute, or field, within the Fire event. In Standard Query Language (SQL) the query would look like this:

```
SELECT Impact.FireEvent, Fire.Fire, Fire.FireCommand,
FireCommand.FireCommand, FireCommand.TargetId, Fire
Command.FirerPlatform
FROM Impact, Fire, FireCommand
WHERE Impact.Fire = 'IE00001'
AND Fire.Fire = Impact.FireEvent
AND FireCommand.FireCommand = Fire.FireCommand
```

The SELECT section chooses the fields the user wants to retrieve and/or compare. The FROM section determines the tables in which to look in for the fields named in the SELECT portion of the SQL statement. WHERE is the comparison portion of the SQL statement specifying the criteria by which records will be eliminated or selected.

The example SQL statement will query the database for the TargetId and FirerPlatform involved in the FireCommand event that produced the Fire event that led to the Impact event designated as 'IE00001'. The statement selects the record in the Impact table with an id equal to 'IE00001'. It then compares this instance's FireEvent attribute with Fire instances until it finds a match. It will then use the FireCommand attribute of that instance of the Fire event to find the matching FireCommand instance and provide the FirerPlatform and TargetId.

A query between tables that lie more than one table away from each other will require that each link between intermediary entities be represented in the SQL statement.

The most important query for this system is also one of the most complicated – time queries. The format used for time data poses a significant problem for comparisons. The data is stored in Military Date-Time fashion. This is expressed as a string concatenation of the day, the time (24-hour), the time zone (Zulu (Z)), the month, and the year. In order to be able to compare these times and to perform any calculations, the string must be segmented into pieces and each portion compared with the appropriate section of another date-time string using string manipulation code.

4. Reports

It is possible to export database data to Microsoft Office applications from a Visual Basic application through the use of automation. Automation is a process that enables applications to communicate and exchange data with each other.

The Visual Basic application sets a reference to the object library of the Automation server to be used; in this case, Word 97®. An instance of the Automation server object is created. A recordset object containing the data desired is also created. A code module within the Visual Basic application instantiates the object variables and sends the data to Word. The data is then exported from the recordset object to the Automation server. Word can then be used to produce and edit a document or merely as a print engine. [Ref. 4] Once the Visual Basic application opens a document in Word 97® the manipulation of data and format of the report is controlled in the Word 97® application. An example of a report produced and formatted in Word 97® is shown in Figure 18.

Information retrieved from the database using predefined queries, SQL custom queries, or data objects (recordsets) can be exported to a report (Word document) by using the Word 97 automation server. Reporting capability is available on forms displaying entity information by means of a menu option.

5. Modules

Visual Basic is an event-driven programming language. Event-driven means that procedures (sections of code intended to perform a singular, specific action) are triggered by user interactions with the application's user interface. User actions can vary from selecting a menu option to selecting a check box in the window. For Visual Basic, modules are groupings of multiple procedures that provide the functionality of a form. Consequently, a module exists for each form in the application. The modules that define the application are provided as Appendix B for this thesis.

Acquisition	TrackId	AcqTime	ThreatType	AcqPlatform	AcqSensorType	TrackLocation	TrackAltitude	TrackSpeed	Track Priority	Remark
AE00036	CSG324 124736		SAM6	CG-71	RPV	324353.7N 0122710.6E		m Knots	TCT	SAM6-CSG324124736
AE00037	GI1745	071421	NODONG	CVN-69		325427.4N 0131517.9E	+0000	m Knots	TCT	3 NODONGS GI1745
AE00038	GI2090	091108	ALABBA	CVN-69	RPV	322430N 0150556E	109	m Knots	TCT	ALABBA-GI2090
AE00039	GI2097	091227	FTR	CVN-69		324918.7N 0125737.4E	57	m Knots	TCT	FUELING TRUCK-GI2097
AE00040	GI2104		SSMDEP	CVN-69		325413.3N 0131111.6E	+0000	m Knots	TCT	SUB-I2104-GI2104
AE00041	GI2111		ALABBA	CVN-69		322433.1N 0150552.0E	+0063	m Knots	TCT	ALABBA-GI2111
AE00042	GI2118		SSMDEP	CVN-69		325402.2N 0131159.9E	6	m Knots	TCT	SUB-GI2118
AE00043	GI2125		CDCM	CVN-69		320000N 0132000E	+0823	m Knots	TCT	LOITER W-GI2125
AE00044	GI0001		SCUDB	JFACC		283500N 0172800E		m Knots	TCT	AL JAFRAH OIL FIELD-G
AE00045	GI0008	070626	SAM2	JFACC	RPV	320639.5N 0200417.0E	+0025	m Knots	TCT	SAM2-GI0008
AE00046	GI0064		NODONG	JFACC		291841.1N 0161827.3E	+0607	m Knots	TCT	NODONG-GI0064
AE00047	GI2001	091104	ALABBA	JFACC	RECO	291719.4N 0173711.1E	806	m Knots	TCT	ALABBAS-GI2001
AE00048	LA5000		NODONG	CG-68		322800N 0115730E	+0075	m Knots	TCT	LA5000 LOITER TTLAM1-
AE00049	LA5010		NODONG	CG-68		290700N 0155730E	+0278	m Knots	TCT	LA5010 LOITER TTLAM2-
AE00050	LG0029		SCUDB	CG-71		283500N 0172800E	+0286	m Knots	TCT	AL JAFRAH OIL FIELD-L

Figure 18 Acquisition Events Report Created in Word 97® by Using Print Option on Acquisition Event Update Form (Figure 10).

C. POST-IMPLEMENTATION

Once the forms were designed and most of the data manipulation code was implemented, the prototype was demonstrated. Feedback provided some new insight into application functionality and some requirement modifications.

Difficult at best, application testing was time-consuming and imprecise. Troubleshooting was an arduous task because of the lack of available of data and the incompleteness of the little data that was available. However, the export facility available in the latest version of LAWS provided some data that could be manipulated, or massaged, into a usable form and used to test the database schema and the interaction of the application with the database.

Forms properly interact with each other and limited error trapping has been included to eliminate database error terminations.

Further discussions with FBE personnel and personal data analysis have demonstrated that the schema developed could be improved by combining some entities such as the LAWS, GISRS, and PTW terminal entities. Data such as time needs to be formatted in a different manner to allow for more effective querying. Other data fields need to be more clearly defined. A lot of fields are defined as text data types, which produces a lot of wasted memory space. This slows the system down and will be apparent once more data is entered.

V. CONCLUSION AND RECOMMENDATION

A. CONCLUSION

The database schema and application developed for this thesis are basic, yet functional. They provide a simple approach for extracting and updating FBE data using an easy to use graphical interface.

The requirements analysis portion of this project was very frustrating. Users were not certain of what capabilities they wanted and had little understanding of the functionality a system such as this could provide. They provided very little input as to desired queries, reports, and interface design preferences. As a result the author had to spend a considerable amount of time understanding the application domain in order to suggest useful query and report options.

The conceptual model should have been more clearly defined and the project's implementation postponed, but time constraints required that the schema be implemented and tested quickly to allow time for the application's development. This led to modifications being applied to the schema mid-way through the implementation of the application.

The Visual Basic and Access coupling is a very effective method of implementing client/server systems such as this because it allows the programmer more control over the design of the interface and its interactions with the database. It also provides for program/data independence thus supporting upward scalability of the database.

However, limitations of relational databases join processing can slow the speed of searches and other data management processes. Numerous table joins are necessary to navigate the database and, with large amounts of data in the tables, querying speed is likely to suffer. We do not however anticipate the volume of data from a single experiment to overwhelm the system since a database contains information from only one experiment.

Another potential limitation is that SQL custom queries require the user to

understand the data model well and can become complicated and slow when information from two or more tables is required.

Perhaps the greatest shortcoming of the Visual Basic and Access linkage is the quantity of error-trapping and data format-validation code that must be created. Data editing and updating method errors addressed by Access 97 can cause an error that may terminate the connection between the application data object and the database.

The database schema was initially implemented in Access 2000, but Visual Basic 6.0 does not seem to interface with Access 2000. This required that the database schema be reconstructed in Access 97.

B. RECOMMENDATION

It is my recommendation that the data model be examined with greater detail. In my opinion, the conceptual model should be researched independently of any implementation plans with the intention of defining data types and sources more clearly. Research should involve input from all data repository systems involved in FBEs, such as LAWS and GISRS, to determine the data overlap of the systems and the distinguishing data interests of each.

The application provides limited sorting and filtering capabilities for the tables defined in the database. The data filtering option needs to be more robust and tested with sufficient feedback from the user. For example, the target record viewing filters are basic and a better format for time data would provide more effective querying capabilities and make the system more stable.

The data "drill-down" functionality is also very basic. Better requirements definition needs to be the main focus of any follow-on work. With a clearer idea of what kind of data is wanted, the system functionality can be more focused and provide better results.

The development effort for this system was implemented using an event-based methodology. Focus was placed on the information and its flow with less attention paid to the behavior of the system. With future alterations to war-fighting doctrine and FBE

architectures, the schema developed here may or may not accurately define the interactions among the entities involved. This schema should be meticulously tested in a FBE environment in order to reveal its limitations.

The fourth phase of the larger effort will use an object-oriented approach. It will look at these relationships as interactions among objects. These objects would contain attributes (information, data) about themselves and behaviors that would define their interactions and information flow and manipulation. By focusing on both information and behavior it is possible to develop systems that are resilient and flexible to changes in structure and processes.

I would also recommend that the system be migrated to the World Wide Web. This would increase the amount of input from interested parties and help define additional requirements. Schema limitations could also be scrutinized and additional entities and relationships added to supplement the data model.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: GLOSSARY OF TERMS

This appendix contains the definitions of abbreviations used in this thesis.

AO	Area of Operations
ANZ	USS ANZIO
ASM	Anti-Ship Missile
ASMD	Anti-Ship Missile Defense
ASW	Anti-Submarine Warfare
ATO	Air Tasking Order
BDA	Bomb Damage Assessment
	Battle Damage Assessment
BWC	Battle Watch Captain
C2	Command and Control
C2W	Command and Control Warfare
C4I	Command, Control, Communications, Computers, and Intelligence
C4ISR	Command, Control, Communications, Computers, Intelligence,
	Surveillance and Reconnaissance
CG	Guided Cruiser
	Commanders Guidance
CIC	Combat Information Center
CINC	Commander-in-Chief
CJTF	Commander Joint Task Force
COA	Course of Action
COMSEC	Communications Security
CONOPS	Concept of Operations
COP	Common Operational Picture
COTS	Commercial Off-the-Shelf
CSG	USS CAPE ST GEORGE
CTF	Combined Task Force

CVBG	Aircraft Carrier Battle Group
CVW	Carrier Air Wing
DFN	Digital Fires Network
DTG	Date Time Group
ERGM	Extended Range Guided Munition
GBS	Global Broadcasting System
GCCS	Global Command and Control System
GIS	Global Information System
GPS	Global Positioning System
GSIRS-C	Global ISR System-Capability
HARM	High Speed Anti-Radiation Missile
HIT	High Interest Track
HOS	Hostile
IKE	USS EISENHOWER
IMMM	In-flight Mission Modification Message
INTEL	Intelligence
ITO	Integrated Task Order
IW	Information Warfare
JAC	Joint Analysis Center
JDAM	Joint Direct Attack Munition
JECG	Joint Experiment Control Group
JFACC	Joint Force Air Component Commander
JFC	Joint Force Commander
JFMCC	Joint Force Maritime Commander
JICO	Joint Interface Control Officer
JIPTL	Joint Integrated Prioritized Target List
JMCIS	Joint Maritime Command Information System
JOA	Joint Operations Area
JSOW	Joint Stand-Off Weapon
JSTARS	Joint Surveillance, Target Attack Radar System

JSWS	JSTARS Work Station
JTCB	Joint Targeting Coordination Board
JTF	Joint Task Force
JTW	Joint Targeting Workstation
JWAC	Joint Warfare Analysis Center
LAMPS	Light Airborne Multipurpose System
LAN	Local Area Network
LANTIRN	Low Altitude Navigation and Targeting Infrared for Night
LAS	USS LASALLE
LASM	Land Attack Standard Missile
LAWS	Land Attack Warfare System
LOS	Line of Sight
LPMP	Launch Point Mission Planner
MBC	Maritime Battle Center
MCC	Maritime Control Center
MDS	Mission Distribution System
METOC	Meteorological and Oceanographic
MOE	Measure of Effectiveness
MPA	Maritime Patrol Aircraft
MPR	Mission Planning Request
NCA	National Command Authority
NLT	Not Later Than
NPS	Naval Post Graduate School
NRO	National Reconnaissance Office
NSFS	Naval Surface Fires Support
NSW	Navy Special Warfare
NUWC	Naval Undersea Warfare Command
NWC	Naval War College
OCE	Officer Conducting Exercise
OIC	Officer-in-Charge

OOB	Order of Battle
OPCON	Operational Control
OPFOR	Opposing Forces
OPINTEL	Operational Intelligence
OPORD	Operations Order
OPSEC	Operational Security
OTCIXS	Officer-in-Tactical Command Information Exchange System
PAO	Public Affairs Office
PGM	Precision Guided Munition
PLI	Position Location Information
POC	Point of Contact
RFI	Request for Information
ROE	Rules of Engagement
SA	Situational Awareness
SAM	Surface-to-Air Missile
SAR	Search and Rescue
	Synthetic Aperture RADAR
SATCOM	Satellite Communications
SC	Strike Controller
SITREP	Situation Report
SLAM	Stand-off Land Attack Missile
SLAM-ER	Stand-off Land Attack Missile Expanded Response
SOF	Special Operations Forces
SOP	Standard Operating Procedure
SPAWARSYSCEN	Space and Naval Warfare Systems Command
SPECWARGRU	Special Warfare Group
SSM	Surface-to-Surface Missile
SSN	Strike Submarine
STOW	Synthetic Theater of War
TA	Target Acquisition

TACAIR	Tactical Aircraft
TARPS	Tactical Air Reconnaissance Pod System
TBD	To Be Determined
TCC	Time Critical Contact
TCT	Time Critical Target
TEL	Transporter Erector Launcher
TLAM	Tomahawk Land Attack Missile
TLE	Target Location Error
TOT	Time on Target
TTF	Time to Fire
TTP	Tactics, Techniques, and Procedures
TST	Time Sensitive Target
TTLAM	Tactical Tomahawk
TTWCS	Tactical Tomahawk Weapons Control System
UAE	Unidentified Assumed Enemy
UAV	Unmanned Aerial Vehicle
USN	United States Navy

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: APPLICATION CODE MODULES

This appendix contains the Visual Basic code modules that define the data management system's interface and functionality.

```
'************************************************************
'Module:         Module1.bas
'Description:    Displays an Open File dialog box upon execution
'                and loads the main form after setting the
'                database file name as a global variable.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Public db                As Database
Public gstNewDatabase    As String
Public gstFBE            As String
Public bContinue         As Boolean
Public bWord             As Boolean
Public bText             As Boolean

Sub Main()

    On Error GoTo HandleError
    gstNewDatabase = GetNewDatabase

    Set db = OpenDatabase(gstNewDatabase)

    'display Main form
    With frmMain
        .Show
        .Caption = .dlgDatabase.FileTitle & " Database"
    End With

Sub_Exit:
    Exit Sub

HandleError:

    Select Case Err.Number
        Case 3004, 3024, 3044
            gstNewDatabase = GetNewDatabase
            If gstNewDatabase = "" Then
                MsgBox "No database was selected.", vbExclamation,
"Database Error"

                'disables options only available when a database is
selected
                frmMain.mnuFileQueries.Enabled = False
                frmMain.mnuFileSQL.Enabled = False
```

```
                frmMain.mnuUpdate.Enabled = False
                Resume Next

            Else
                Set db = OpenDatabase(gstNewDatabase)  'new database
location

                'reenables options once a database is selected
                frmMain.mnuFileQueries.Enabled = True
                frmMain.mnuFileSQL.Enabled = True
                frmMain.mnuUpdate.Enabled = True

                Resume      'open the database

            End If
        Case Else
            MsgBox Err.Description, vbOKOnly + vbExclamation,
"Unexpected Error"
            End             'exit the project

    End Select

End Sub

Public Function GetNewDatabase() As String
    'allows user to browse for database
    Dim iResp   As Integer
    Dim stMsg   As String

    stMsg = "Do you want to locate a database file?"

    iResp = MsgBox(stMsg, vbYesNo + vbQuestion, "File or Path not
found")

    'if user does not want to browse for file
    If iResp = vbNo Then

        'set database name to "blank"
        GetNewDatabase = ""

    Else
        'if user opts to find database
        With frmMain.dlgDatabase
            .FileName = App.Path & "\G.mdb"  'gstNewDatabase
            .Filter = "Database files (*.mdb)|*.mdb|All files
(*.*)|*.*"

            'if error encountered, skip next command
            On Error Resume Next
            .ShowOpen
            If Err.Number = cdlCancel Then
                GetNewDatabase = ""
            Else

                'set return filename to selected file
```

```
                GetNewDatabase = .FileName
            End If
        End With
    End If

End Function


'*************************************************************
'Module:        frmAbout.frm
'Programmer:    Kevin Colón
'*************************************************************

Option Explicit

' Reg Key Security Options...
Const READ_CONTROL = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE +
KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + KEY_NOTIFY +
KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                        ' Unicode nul terminated
string
Const REG_DWORD = 4                     ' 32-bit number

Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias
"RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal
ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long)
As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias
"RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String,
ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String,
ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As
Long) As Long


Private Sub cmdSysInfo_Click()
  Call StartSysInfo
End Sub
```

```
Private Sub cmdOK_Click()
  Unload Me
End Sub

Private Sub Form_Load()
    Me.Caption = "About " & App.Title
    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "."
& App.Revision
    lblTitle.Caption = App.Title
End Sub

Public Sub StartSysInfo()
    On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String

    ' Try To Get System Info Program Path\Name From Registry...
    If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO,
SysInfoPath) Then
    ' Try To Get System Info Program Path Only From Registry...
    ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC,
gREGVALSYSINFOLOC, SysInfoPath) Then
        ' Validate Existance Of Known 32 Bit File Version
        If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
            SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

        ' Error - File Can Not Be Found...
        Else
            GoTo SysInfoErr
        End If
    ' Error - Registry Entry Can Not Be Found...
    Else
        GoTo SysInfoErr
    End If

    Call Shell(SysInfoPath, vbNormalFocus)

    Exit Sub
SysInfoErr:
    MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub

Public Function GetKeyValue(KeyRoot As Long, KeyName As String,
SubKeyRef As String, ByRef KeyVal As String) As Boolean
    Dim i As Long                                           ' Loop
Counter
    Dim rc As Long                                          ' Return
Code
    Dim hKey As Long                                        ' Handle To
An Open Registry Key
    Dim hDepth As Long                                      '
    Dim KeyValType As Long                                  ' Data Type
Of A Registry Key
    Dim tmpVal As String                                    ' Tempory
```

```
Storage For A Registry Key Value
    Dim KeyValSize As Long                                  ' Size Of
Registry Key Variable
    '------------------------------------------------------------
    ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
    '------------------------------------------------------------
    rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open
Registry Key

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle
Error...

    tmpVal = String$(1024, 0)                              ' Allocate
Variable Space
    KeyValSize = 1024                                       ' Mark
Variable Size

    '------------------------------------------------------------
    ' Retrieve Registry Key Value...
    '------------------------------------------------------------
    rc = RegQueryValueEx(hKey, SubKeyRef, 0, _
                         KeyValType, tmpVal, KeyValSize)    '
Get/Create Key Value

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle
Errors

    If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then           ' Win95
Adds Null Terminated String...
        tmpVal = Left(tmpVal, KeyValSize - 1)               ' Null
Found, Extract From String
    Else                                                    ' WinNT
Does NOT Null Terminate String...
        tmpVal = Left(tmpVal, KeyValSize)                   ' Null Not
Found, Extract String Only
    End If
    '------------------------------------------------------------
    ' Determine Key Value Type For Conversion...
    '------------------------------------------------------------
    Select Case KeyValType                                  ' Search
Data Types...
    Case REG_SZ                                             ' String
Registry Key Data Type
        KeyVal = tmpVal                                     ' Copy
String Value
    Case REG_DWORD                                          ' Double
Word Registry Key Data Type
        For i = Len(tmpVal) To 1 Step -1                    ' Convert
Each Bit
            KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))   ' Build
Value Char. By Char.
        Next
        KeyVal = Format$("&h" + KeyVal)                     ' Convert
Double Word To String
    End Select
```

```
    GetKeyValue = True                                          ' Return
Success
    rc = RegCloseKey(hKey)                                      ' Close
Registry Key
    Exit Function                                               ' Exit

GetKeyError:      ' Cleanup After An Error Has Occured...
    KeyVal = ""                                                 ' Set
Return Val To Empty String
    GetKeyValue = False                                         ' Return
Failure
    rc = RegCloseKey(hKey)                                      ' Close
Registry Key
End Function

Private Sub Form_Unload(Cancel As Integer)

    Unload Me

End Sub


'*************************************************************
'Module:         frmAcqEvents.frm
'Description:    Allows user to access acquisition event
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

Private WordApp     As Word.Application
Private Doc         As Word.Document
Private Sel         As Word.Selection

Private Sub cboPlatforms_Click()

    If cboPlatforms.ListIndex >= 0 Then
        txtPlatformId = cboPlatforms.Text

    End If


End Sub

Private Sub cboSensTypes_Click()

    If cboSensTypes.ListIndex >= 0 Then
        txtSensorTypeId = cboSensTypes.Text

    End If

End Sub
```

```
Private Sub cboThreatTypes_Click()

    If cboThreatTypes.ListIndex >= 0 Then
        txtThreatTypeId = cboThreatTypes.Text

    End If

End Sub

Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add Event" Then

        datAcqEvents.Recordset.AddNew
        cboPlatforms.Enabled = True
        cboPlatforms.ListIndex = -1
        cboThreatTypes.Enabled = True
        cboThreatTypes.ListIndex = -1
        cboSensTypes.Enabled = True
        cboSensTypes.ListIndex = -1
        txtTrackId = True
        txtAssess.Enabled = True
        txtAltitude.Enabled = True
        txtLocation.Enabled = True
        txtPriority.Enabled = True
        txtSpeed.Enabled = True
        txtTime.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datAcqEvents.Enabled = False

    Else
        datAcqEvents.Recordset.CancelUpdate
        datAcqEvents.Enabled = True
        cboPlatforms.Enabled = False
        cboThreatTypes.Enabled = False
        cboSensTypes.Enabled = False
        txtTrackId = False
        txtAssess.Enabled = False
        txtAltitude.Enabled = False
        txtLocation.Enabled = False
        txtPriority.Enabled = False
        txtSpeed.Enabled = False
        txtTime.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
```

```
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping


End Sub

Private Sub cmdDel_Click()

    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datAcqEvents.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Event " & txtAcqEventId & "?", vbYesNo,
_
                "Delete Event")
        If iResp = vbYes Then
            With datAcqEvents.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Event"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
```

```
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdSave_Click()

    'save the current record

    On Error GoTo HandleSaveErrors
    If cboThreatTypes.ListIndex >= 0 And cboSensTypes.ListIndex >= 0
Then
        If Val(txtCounter) < 10 Then
            txtAcqEventId.Text = "AE0000" & txtCounter.Text
        Else
            If Val(txtCounter) < 100 Then
                txtAcqEventId.Text = "AE000" & txtCounter.Text
            Else
                If Val(txtCounter) < 1000 Then
                    txtAcqEventId.Text = "AE00" & txtCounter.Text
                Else
                    If Val(txtCounter) < 10000 Then
                        txtAcqEventId.Text = "AE0" & txtCounter.Text
                    Else
                        txtAcqEventId.Text = "AE" & txtCounter.Text
                    End If
                End If
            End If
        End If

        datAcqEvents.Recordset.Update
    Else
        MsgBox "You must select a threat and sensor before saving." _
                , vbExclamation, "Add Acquisition Event"
        datAcqEvents.Recordset.CancelUpdate
    End If

    cboPlatforms.Enabled = False
    cboThreatTypes.Enabled = False
    cboSensTypes.Enabled = False
    txtTrackId = False
    txtAssess.Enabled = False
    txtAltitude.Enabled = False
    txtLocation.Enabled = False
    txtPriority.Enabled = False
    txtSpeed.Enabled = False
    txtTime.Enabled = False
    cmdSave.Enabled = False
    cmdUpdate.Enabled = True
    cmdDel.Enabled = True
    cmdAdd.Caption = "&Add Event"
    mnuFile.Enabled = True
    datAcqEvents.Enabled = True
    cmdAdd.SetFocus
```

```
    datAcqEvents.Enabled = True

cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Select a Sensor type and threat before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datAcqEvents.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()

    If cmdUpdate.Caption = "&Update" And _
        datAcqEvents.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboPlatforms.Enabled = True
        cboThreatTypes.Enabled = True
        cboSensTypes.Enabled = True
        txtTrackId = True
        txtAssess.Enabled = True
        txtAltitude.Enabled = True
        txtLocation.Enabled = True
        txtPriority.Enabled = True
        txtSpeed.Enabled = True
        txtTime.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        cmdAdd.Enabled = False
        mnuFile.Enabled = False
        datAcqEvents.Enabled = False
        datAcqEvents.Recordset.Edit

    Else
        If datAcqEvents.Recordset.RecordCount > 0 Then
            datAcqEvents.Recordset.Update
```

```
            cboPlatforms.Enabled = False
            cboThreatTypes.Enabled = False
            cboSensTypes.Enabled = False
            txtTrackId = False
            txtAssess.Enabled = False
            txtAltitude.Enabled = False
            txtLocation.Enabled = False
            txtPriority.Enabled = False
            txtSpeed.Enabled = False
            txtTime.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datAcqEvents.Enabled = True

        End If
    End If

End Sub

Private Sub datAcqEvents_Reposition()

    SetAcqEventsRecordNumber

End Sub


Private Sub FillPlatformCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
    cboPlatforms.Clear

    With datPlatforms
        .Refresh        'open database
        iCount = .Recordset.RecordCount

        'fill the list
        Do Until .Recordset.EOF
            If .Recordset!Platform <> "" Then
                cboPlatforms.AddItem .Recordset!Platform

            End If
            .Recordset.MoveNext
        Loop
    End With

End Sub

Private Sub FillSensTypeCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
```

```
    cboSensTypes.Clear

    With datSensTypes
        .Refresh        'open database

        Do Until .Recordset.EOF     'fill the list
            If .Recordset!SensorType <> "" Then
                cboSensTypes.AddItem .Recordset!SensorType

            End If
            .Recordset.MoveNext

        Loop
    End With

End Sub

Private Sub FillThreatTypeCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
    cboThreatTypes.Clear

    With datThreatTypes
        .Refresh        'open database

        Do Until .Recordset.EOF     'fill the list
            If .Recordset!ThreatType <> "" Then
                cboThreatTypes.AddItem .Recordset!ThreatType

            End If
            .Recordset.MoveNext

        Loop
    End With

End Sub


Private Sub Form_Load()

    datPlatforms.DatabaseName = gstNewDatabase
    datSensTypes.DatabaseName = gstNewDatabase
    datThreatTypes.DatabaseName = gstNewDatabase
    datAcqEvents.DatabaseName = gstNewDatabase

    FillPlatformCombo
    FillSensTypeCombo
    FillThreatTypeCombo

    With datAcqEvents
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
```

```
        End If
    End With

    SetAcqEventsRecordNumber


End Sub

Private Sub SetAcqEventsRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datAcqEvents.Recordset.RecordCount
    iCurrentRecord = datAcqEvents.Recordset.AbsolutePosition + 1

    If datAcqEvents.Recordset.EOF Then
        datAcqEvents.Caption = "No more records"
    Else
        datAcqEvents.Caption = "Acquisition Event Record " &
iCurrentRecord & _
                            " of " & iRecordCount
    End If

End Sub


Private Sub mnuFileBack_Click()

    frmMain.Show
    frmMain.Enabled = True
    Unload Me


End Sub

Private Sub mnuPrint_Click()

    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datAcqEvents.Recordset

            If bWord = True Then

                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=11
```

```
                Sel.TypeText Text:="Acquisition"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="TrackId"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="AcqTime"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="ThreatType"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="AcqPlatform"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="AcqSensorType"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="TrackLocation"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="TrackAltitude"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="TrackSpeed"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="TrackPriority"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Remark"
                Sel.MoveRight unit:=12                  '12=next cell

                Do Until .EOF

                    Sel.TypeText Text:=!Acquisition
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TrackId
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!AcqTime
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!ThreatType
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!AcqPlatform
                    Sel.MoveRight unit:=12                  '12=next
cell
```

```
                    Sel.TypeText Text:=!AcqSensorType
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TrackLocation
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TrackAltitude
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TrackSpeed
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TrackPriority
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!Remark
                    Sel.MoveRight unit:=12                  '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\AcqEvents.txt" For Output As #1

                    Print #1, "Acquisition"; Chr(9); "TrackId"; Chr(9);
"AcqTime"; Chr(9); _
                                "ThreatType"; Chr(9); "AcqPlatform";
Chr(9); _
                                "AcqSensorType"; Chr(9);
"TrackLocation"; Chr(9); _
                                "TrackAltitude"; Chr(9); "TrackSpeed";
Chr(9); _
                                "TrackPriority"; Chr(9); "Remark";
Chr(9)

                    Do Until .EOF

                        Print #1, !Acquisition; Chr(9); _
                                  !TrackId; Chr(9); _
                                  !AcqTime; Chr(9); _
```

```
                                    !ThreatType; Chr(9); _
                                    !AcqPlatform; Chr(9); _
                                    !AcqSensorType; Chr(9); _
                                    !TrackLocation; Chr(9); _
                                    !TrackAltitude; Chr(9); _
                                    !TrackSpeed; Chr(9); _
                                    !TrackPriority; Chr(9); _
                                    !Remark; Chr(9)

                        .MoveNext
                    Loop

                    Close #1

                End If
            End If

        .MoveFirst

        End With

    End If

    bContinue = False
    bWord = False
    bText = False


mnuPrintErrors:
        Select Case Err.Number
            Case 94
                Sel.TypeText Text:=""
                Resume Next
        End Select

End Sub

Private Sub txtPlatformId_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
    Dim bFound  As Boolean

    datPlatforms.Recordset.MoveFirst
    If txtPlatformId <> "" Then
        Do Until iIndex = datPlatforms.Recordset.RecordCount Or bFound
            If datPlatforms.Recordset!Platform = txtPlatformId Then
                cboPlatforms.Text = datPlatforms.Recordset!Platform
                bFound = True
            Else
                datPlatforms.Recordset.MoveNext
                iIndex = iIndex + 1

            End If
        Loop
```

```
    Else
        cboPlatforms.ListIndex = -1
    End If

End Sub

Private Sub txtSensorTypeId_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
    Dim bFound  As Boolean
    Dim num As Integer

    With datSensTypes
        .Recordset.MoveFirst
        If txtSensorTypeId <> "" Then

            Do Until iIndex = cboSensTypes.ListCount Or bFound
                If .Recordset!SensorType = txtSensorTypeId Then
                    cboSensTypes.Text = .Recordset!SensorType
                    bFound = True
                Else
                    .Recordset.MoveNext
                    iIndex = iIndex + 1

                End If
            Loop
        Else
            cboSensTypes.ListIndex = -1

        End If
    End With


End Sub

Private Sub txtThreatTypeId_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
    Dim bFound  As Boolean

    datThreatTypes.Recordset.MoveFirst
    If txtThreatTypeId <> "" Then

        Do Until iIndex = cboThreatTypes.ListCount Or bFound
            If datThreatTypes.Recordset!ThreatType = txtThreatTypeId
Then
                cboThreatTypes.Text =
datThreatTypes.Recordset!ThreatType
                bFound = True
            Else
                datThreatTypes.Recordset.MoveNext
                iIndex = iIndex + 1
```

```
            End If
        Loop
    Else
        cboThreatTypes.ListIndex = -1

    End If

End Sub


'************************************************************
'Module:         frmAcronyms.frm
'Description:    Allows user to access acronyms
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Private Sub cmdAddAcronym_Click()
    On Error GoTo HandleAddAcronymErrors

    If cmdAddAcronym.Caption = "&Add Acronym" Then
        datAcronyms.Recordset.AddNew
        txtAcronym.Enabled = True
        txtAcronym.SetFocus
        txtDescription.Enabled = True
        cmdAddAcronym.Caption = "&Cancel"
        cmdSaveAcronym.Enabled = True
        cmdDelAcronym.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datAcronyms.Enabled = False

    Else
        datAcronyms.Recordset.CancelUpdate
        txtAcronym.Enabled = False
        txtDescription.Enabled = False
        cmdSaveAcronym.Enabled = False
        cmdDelAcronym.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAddAcronym.Caption = "&Add Acronym"
        cmdAddAcronym.SetFocus
        datAcronyms.Enabled = True

    End If

cmdAddAcronym_Click_Exit:
    Exit Sub

HandleAddAcronymErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
```

```
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping


End Sub

Private Sub cmdDelAcronym_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelAcronymErrors

    If datAcronyms.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Acronym " & txtAcronym.Text & "?",
vbYesNo, "Delete Acronym")
        If iResp = vbYes Then
            With datAcronyms.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                       MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Acronym"

    End If

cmdDelAcronym_Click_Exit:
    Exit Sub

HandleDelAcronymErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdSaveAcronym_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveAcronymErrors
    If txtAcronym <> "" And txtDescription <> "" Then
        txtAcronym = UCase(txtAcronym)
```

```
        iResp = MsgBox("Do you want to add " & txtAcronym & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Acronym")
        If iResp = vbYes Then
            datAcronyms.Recordset.Update
        End If

    Else
        MsgBox "You must enter an acronym and a description before
saving.", vbExclamation _
                , "Add Acronym"
        datAcronyms.Recordset.CancelUpdate
    End If

    txtAcronym.Enabled = False
    txtDescription.Enabled = False
    cmdSaveAcronym.Enabled = False
    cmdDelAcronym.Enabled = True
    datAcronyms.Enabled = True
    mnuFile.Enabled = True
    cmdAddAcronym.Caption = "&Add Acronym"
    cmdAddAcronym.SetFocus
    cmdUpdate.Enabled = True


cmdSaveAcronym_Click_Exit:
    Exit Sub

HandleSaveAcronymErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a Acronym name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datAcronyms.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datAcronyms.Recordset.RecordCount > 0 Then
```

```
        cmdUpdate.Caption = "Su&bmit"
        txtAcronym.Enabled = True
        txtDescription.Enabled = True
        cmdDelAcronym.Enabled = False
        mnuFile.Enabled = False
        txtAcronym.SetFocus
        cmdAddAcronym.Enabled = False
        datAcronyms.Enabled = False
        datAcronyms.Recordset.Edit
    Else
        If datAcronyms.Recordset.RecordCount > 0 Then
            datAcronyms.Recordset.Update

            txtAcronym.Enabled = False
            txtDescription.Enabled = False
            cmdDelAcronym.Enabled = True
            mnuFile.Enabled = True
            cmdAddAcronym.Enabled = True
            cmdAddAcronym.SetFocus
            cmdUpdate.Caption = "&Update"
            datAcronyms.Enabled = True
        End If
    End If

End Sub

Private Sub datAcronyms_Reposition()
    SetAcronymRecordNumber

End Sub

Private Sub Form_Load()
    datAcronyms.DatabaseName = gstNewDatabase

    With datAcronyms
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetAcronymRecordNumber
'

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()
```

```
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datAcronyms.Recordset.FindFirst "[Acronym] = '" & _
                InputBox("Enter the Acronym", "Acronym Search") & "'"

    If datAcronyms.Recordset.NoMatch Then
        MsgBox "Acronym was not found.", vbOKOnly, "Acronym Search"
        datAcronyms.Recordset.MoveFirst          'go to first record
    End If

End Sub

Private Sub SetAcronymRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datAcronyms.Recordset.RecordCount
    iCurrentRecord = datAcronyms.Recordset.AbsolutePosition + 1
    If datAcronyms.Recordset.EOF Then
        datAcronyms.Caption = "No more records"
    Else
        datAcronyms.Caption = "Acronym " & iCurrentRecord & _
                              " of " & iRecordCount
    End If

End Sub


'************************************************************
'Module:         frmDataTypes.frm
'Description:    Allows user to access the data types
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datDataTypes.Recordset.AddNew
        txtDataType.Enabled = True
        txtDataType.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
```

```
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datDataTypes.Enabled = False

    Else
        datDataTypes.Recordset.CancelUpdate
        txtDataType.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datDataTypes.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping


End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datDataTypes.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete DataType " & txtDataType.Text & "?",
vbYesNo, "Delete DataType")
        If iResp = vbYes Then
            With datDataTypes.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
```

```
                , "Delete DataType"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtDataType <> "" And txtDescription <> "" Then
        txtDataType = UCase(txtDataType)
        iResp = MsgBox("Do you want to add " & txtDataType & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add DataType")
        If iResp = vbYes Then
            datDataTypes.Recordset.Update
        End If

    Else
        MsgBox "You must enter an DataType and a description before
saving.", vbExclamation _
                , "Add DataType"
        datDataTypes.Recordset.CancelUpdate
    End If

    txtDataType.Enabled = False
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datDataTypes.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
```

```
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a DataType name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datDataTypes.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datDataTypes.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtDataType.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtDataType.SetFocus
        cmdAdd.Enabled = False
        datDataTypes.Enabled = False
        datDataTypes.Recordset.Edit
    Else
        If datDataTypes.Recordset.RecordCount > 0 Then
            datDataTypes.Recordset.Update

            txtDataType.Enabled = False
            txtDescription.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datDataTypes.Enabled = True
        End If
    End If

End Sub

Private Sub datDataTypes_Reposition()
    SetDataTypeRecordNumber

End Sub
```

```
Private Sub Form_Load()
    datDataTypes.DatabaseName = gstNewDatabase

    With datDataTypes
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetDataTypeRecordNumber
'

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datDataTypes.Recordset.FindFirst "[DataType] = '" & _
                InputBox("Enter the Data Type", "Data Type Search") &
"'"

    If datDataTypes.Recordset.NoMatch Then
        MsgBox "Data Type was not found.", vbOKOnly, "Data Type Search"
        datDataTypes.Recordset.MoveFirst         'go to first record
    End If


End Sub

Private Sub SetDataTypeRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datDataTypes.Recordset.RecordCount
    iCurrentRecord = datDataTypes.Recordset.AbsolutePosition + 1
    If datDataTypes.Recordset.EOF Then
        datDataTypes.Caption = "No more records"
    Else
        datDataTypes.Caption = "DataType " & iCurrentRecord & _
                               " of " & iRecordCount
```

```
    End If

End Sub

'************************************************************
'Module:         frmFBE.frm
'Description:    Allows user to access the FBE information
'                for addition, deletion, and modification.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Private Sub cmdAddFBE_Click()
    On Error GoTo HandleAddFBEErrors

    If cmdAddFBE.Caption = "&Add FBE" Then
        datFBE.Recordset.AddNew
        txtFBE.Enabled = True
        txtFBE.SetFocus
        txtDescription.Enabled = True
        txtStart.Enabled = True
        txtEnd.Enabled = True
        cmdAddFBE.Caption = "&Cancel"
        cmdSaveFBE.Enabled = True
        cmdDelFBE.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datFBE.Enabled = False

    Else
        datFBE.Recordset.CancelUpdate
        txtFBE.Enabled = False
        txtDescription.Enabled = False
        txtStart.Enabled = False
        txtEnd.Enabled = False
        cmdSaveFBE.Enabled = False
        cmdDelFBE.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAddFBE.Caption = "&Add FBE"
        cmdAddFBE.SetFocus
        datFBE.Enabled = True

    End If

cmdAddFBE_Click_Exit:
    Exit Sub

HandleAddFBEErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping
```

```
End Sub

Private Sub cmdDelFBE_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelFBEErrors

    If datFBE.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete FBE " & txtFBE.Text & "?", vbYesNo,
"Delete FBE")
        If iResp = vbYes Then
            With datFBE.Recordset
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete FBE"

    End If

cmdDelFBE_Click_Exit:
    Exit Sub

HandleDelFBEErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSaveFBE_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveFBEErrors
    If txtFBE <> "" And txtDescription <> "" And txtStart <> "" And _
        txtEnd <> "" Then
        txtFBE = UCase(txtFBE)
        iResp = MsgBox("Do you want to add " & txtFBE & _
                    " to the database?", vbYesNo + vbQuestion, _
```

```
                    "Add FBE")
        If iResp = vbYes Then
            datFBE.Recordset.Update
        End If

    Else
        MsgBox "You must enter an FBE, a description, and dates before
saving.", vbExclamation _
                , "Add FBE"
        datFBE.Recordset.CancelUpdate
    End If

    txtFBE.Enabled = False
    txtDescription.Enabled = False
    txtStart.Enabled = False
    txtEnd.Enabled = False
    cmdSaveFBE.Enabled = False
    cmdDelFBE.Enabled = True
    datFBE.Enabled = True
    mnuFile.Enabled = True
    cmdAddFBE.Caption = "&Add FBE"
    cmdAddFBE.SetFocus
    cmdUpdate.Enabled = True


cmdSaveFBE_Click_Exit:
    Exit Sub

HandleSaveFBEErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a FBE name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datFBE.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datFBE.Recordset.RecordCount > 0 Then
```

```
        cmdUpdate.Caption = "Su&bmit"
        txtFBE.Enabled = True
        txtDescription.Enabled = True
        txtStart.Enabled = True
        txtEnd.Enabled = True
        cmdDelFBE.Enabled = False
        mnuFile.Enabled = False
        txtFBE.SetFocus
        cmdAddFBE.Enabled = False
        datFBE.Enabled = False
        datFBE.Recordset.Edit
    Else
        If datFBE.Recordset.RecordCount > 0 Then
            datFBE.Recordset.Update

            txtFBE.Enabled = False
            txtDescription.Enabled = False
            txtStart.Enabled = False
            txtEnd.Enabled = False
            cmdDelFBE.Enabled = True
            mnuFile.Enabled = True
            cmdAddFBE.Enabled = True
            cmdAddFBE.SetFocus
            cmdUpdate.Caption = "&Update"
            datFBE.Enabled = True
        End If
    End If

End Sub

Private Sub datFBE_Reposition()

    SetFBERecordNumber

End Sub

Private Sub Form_Load()
    datFBE.DatabaseName = gstNewDatabase

    With datFBE
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetFBERecordNumber

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Show
    frmMain.Enabled = True
```

```
    Unload Me

End Sub

Private Sub mnuFileBack_Click()
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub SetFBERecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datFBE.Recordset.RecordCount
    iCurrentRecord = datFBE.Recordset.AbsolutePosition + 1
    If datFBE.Recordset.EOF Then
        datFBE.Caption = "No more records"
    Else
        datFBE.Caption = "FBE " & iCurrentRecord & _
                         " of " & iRecordCount
    End If

End Sub


'**************************************************************
'Module:        frmFilters.frm
'Description:   Allows user to select filters applied to
'               target recordset parameters.
'Programmer:    Kevin Colón
'**************************************************************

Option Explicit
    Dim rsWeapons       As Recordset
    Dim stSQL           As String
    Dim stSQL1          As String
    Dim stDesigTime     As String
    Dim stDesigDay      As String
    Dim stNLTTime       As String
    Dim stNLTDay        As String
    Dim stLatDeg        As String
    Dim stLatDir        As String
    Dim stLongDeg       As String
    Dim stLongDir       As String


Private Sub chkDescription_Click()

    If chkDescription.Value = 1 Then
        fraDescription.Enabled = True
    Else
        fraDescription.Enabled = False
```

```
    End If

End Sub

Private Sub chkLocation_Click()

    If chkLocation.Value = 1 Then
        fraLocation.Enabled = True
    Else
        fraLocation.Enabled = False
    End If

End Sub

Private Sub chkTime_Click()

    If chkTime.Value = 1 Then
        fraTime.Enabled = True
    Else
        fraTime.Enabled = False
    End If


End Sub

Private Sub chkWeapon_Click()

    If chkWeapon.Value = 1 Then
        fraWeapon.Enabled = True
    Else
        fraWeapon.Enabled = False
    End If

End Sub

Private Sub cmdApply_Click()

    If chkTime = 1 And chkDescription = 1 And chkWeapon = 1 _
        And chkLocation = 1 Then


    Else
        If chkTime = 1 And chkDescription = 1 And chkWeapon = 1 _
            And chkLocation = 0 Then

        Else
            If chkTime = 1 And chkDescription = 1 And chkWeapon = 0 _
                And chkLocation = 1 Then


            Else
                If chkTime = 1 And chkDescription = 0 And chkWeapon = 1
_
                    And chkLocation = 1 Then
```

```
                Else
                    If chkTime = 0 And chkDescription = 1 And chkWeapon
= 1 _
                        And chkLocation = 1 Then


                    Else
                        If chkTime = 1 And chkDescription = 1 And
chkWeapon = 0 _
                            And chkLocation = 0 Then



                        Else
                            If chkTime = 1 And chkDescription = 0 And
chkWeapon = 1 _
                                And chkLocation = 0 Then


                            Else
                                If chkTime = 0 And chkDescription = 1
And chkWeapon = 1 _
                                    And chkLocation = 0 Then

                                    stSQL = "Select * from Target " & _
                                            "Where Description = '" &
txtDescription.Text & "' " & _
                                            "And WeaponType = '" &
cboWeapon.Text & "'"

                                    With frmTargets2.datTargets
                                        .RecordSource = stSQL
                                        .Refresh
                                    End With

                                Else
                                    If chkTime = 1 And chkDescription =
0 And chkWeapon = 0 _
                                        And chkLocation = 1 Then

                                    Else
                                        If chkTime = 0 And
chkDescription = 1 And chkWeapon = 0 _
                                            And chkLocation = 1 Then


                                        Else
                                            If chkTime = 0 And
chkDescription = 0 And chkWeapon = 1 _
                                                And chkLocation = 1
Then

                                            Else
                                                If chkTime = 1 And
chkDescription = 0 And chkWeapon = 0 _
```

```
                                                    And chkLocation = 0
Then

                                                Else
                                                    If chkTime = 0 And
chkDescription = 1 And chkWeapon = 0 _
                                                        And chkLocation
= 0 Then

                                                        stSQL = "Select
* from Target " & _
                                                                "Where
Description = '" & txtDescription.Text & "'"

                                                        With
frmTargets2.datTargets

.RecordSource = stSQL
                                                            .Refresh
                                                        End With

                                                    Else
                                                        If chkTime = 0
And chkDescription = 0 And chkWeapon = 1 _
                                                            And
chkLocation = 0 Then

                                                            stSQL =
"Select * from Target " & _

"Where WeaponType = '" & cboWeapon.Text & "'"

                                                            With
frmTargets2.datTargets

.RecordSource = stSQL

.Refresh
                                                            End With

                                                        Else
                                                            If chkTime
= 0 And chkDescription = 0 And chkWeapon = 0 _
                                                                And
chkLocation = 1 Then


                                                            Else
                                                                If
chkTime = 0 And chkDescription = 0 And chkWeapon = 0 _
                                                                    And
chkLocation = 0 Then
```

```
stSQL = "Select * from Target"

With frmTargets2.datTargets

.RecordSource = stSQL

.Refresh
                                                                    End
With
                                                                End If
                                                            End If
                                                        End If
                                                    End If
                                                End If
                                            End If
                                        End If
                                    End If
                                End If
                            End If
                        End If
                    End If
                End If
            End If
        End If
    End If



    If chkTime.Value = 1 Then

    Else

    End If

    If chkDescription.Value = 1 Then

    Else

    End If

    If chkWeapon.Value = 1 Then
        stSQL = "Select * from Target where WeaponType = '" &
cboWeapon.Text & "'"

        With frmTargets2.datTargets
            .RecordSource = stSQL
            .Refresh
        End With

    Else
        stSQL = "Select * from Target"
```

```
        With frmTargets2.datTargets
            .RecordSource = stSQL
            .Refresh
        End With


    End If

    If chkLocation.Value = 1 Then

    Else

    End If


End Sub

Private Sub cmdCancel_Click()

    frmTargets2.Enabled = True
    Me.Hide

End Sub

Private Sub cmdOK_Click()

    cmdApply_Click
    frmTargets2.Enabled = True
    frmTargets2.Show
    Me.Hide

End Sub

Private Sub FillcboWeapons()

    With rsWeapons
        Do Until .EOF
            cboWeapon.AddItem !WeaponType
            .MoveNext
        Loop
    End With

    cboWeapon.ListIndex = -1


End Sub

Private Sub Form_Load()

    stSQL1 = "Select WeaponType from WeaponType"

    Set rsWeapons = db.OpenRecordset(stSQL1)
```

```
    FillcboWeapons

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmTargets2.Enabled = True
    Me.Hide


End Sub


'*************************************************************
'Module:         frmFireCmdEvent.frm
'Description:    Allows user to access the fire command event
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

Dim rsNomination    As Recordset
Dim rsTarget        As Recordset
Dim rsPlatform      As Recordset
Dim stSQL1          As String
Dim stSQL2          As String
Dim stSQL3          As String
Private WordApp     As Word.Application
Private Doc         As Word.Document
Private Sel         As Word.Selection


Private Sub cboNomination_Change()

    If cboNomination.ListIndex >= 0 Then
        txtNomination = cboNomination.Text
    End If

End Sub

Private Sub cboPlatform_Change()

    If cboPlatform.ListIndex >= 0 Then
        txtPlatform = cboPlatform.Text
    End If

End Sub

Private Sub cboTarget_Change()

    If cboTarget.ListIndex >= 0 Then
```

```
        txtTarget = cboTarget.Text
    End If

End Sub

Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add Event" Then

        datFireCommand.Recordset.AddNew
        cboTarget.Enabled = True
        cboNomination.Enabled = True
        cboPlatform.Enabled = True
        cboTarget.ListIndex = -1
        cboNomination.ListIndex = -1
        cboPlatform.ListIndex = -1
        txtTimeSent.Enabled = True
        txtTimeRcvd.Enabled = True
        txtOCCCId.Enabled = True
        chkEngage.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datFireCommand.Enabled = False

    Else

        datFireCommand.Recordset.CancelUpdate
        cboTarget.Enabled = False
        cboNomination.Enabled = False
        cboPlatform.Enabled = False
        txtTimeSent.Enabled = False
        txtTimeRcvd.Enabled = False
        txtOCCCId.Enabled = False
        chkEngage.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
        datFireCommand.Enabled = True
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess      As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf &
```

```
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0     'turn off error trapping

End Sub

Private Sub cmdDel_Click()

    Dim iResp       As Integer

    On Error GoTo HandleDelErrors

    If datFireCommand.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Event " & txtFireCommand & "?", vbYesNo,
_
                    "Delete Event")
        If iResp = vbYes Then
            With datFireCommand.Recordset
                .Delete
                .MoveNext
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation, "Delete Event"

    End If

cmdDel_Click:
    Exit Sub

HandleDelErrors:
    Dim stMess       As String

    stMess = "Cannot complete operation." & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0

End Sub

Private Sub cmdSave_Click()

    'save current record
    On Error GoTo HandleSaveErrors
    If cboTarget.ListIndex >= 0 And cboNomination.ListIndex >= 0 And
cboPlatform.ListIndex >= 0 Then
        If Val(txtCounter) < 10 Then
            txtFireCommand.Text = "FC0000" & txtCounter.Text
```

```
        Else
            If Val(txtCounter) < 100 Then
                txtFireCommand.Text = "FC000" & txtCounter.Text
            Else
                If Val(txtCounter) < 1000 Then
                    txtFireCommand.Text = "FC00" & txtCounter.Text
                Else
                    If Val(txtCounter) < 10000 Then
                        txtFireCommand.Text = "FC0" & txtCounter.Text
                    Else
                        txtFireCommand.Text = "FC" & txtCounter.Text
                    End If
                End If
            End If
        End If

        datFireCommand.Recordset.Update
    Else
        MsgBox "You must select a Nomination Event, a Target, and a
Platform before saving." _
                , vbExclamation, "Add Fire Command Event"
        datFireCommand.Recordset.CancelUpdate
    End If

    cboTarget.Enabled = False
    cboNomination.Enabled = False
    cboPlatform.Enabled = False
    txtTimeSent.Enabled = False
    txtTimeRcvd.Enabled = False
    txtOCCCId.Enabled = False
    chkEngage.Enabled = False
    cmdUpdate.Enabled = True
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    cmdAdd.Caption = "&Add Event"
    mnuFile.Enabled = True
    datFireCommand.Enabled = True
    cmdAdd.SetFocus


    datFireCommand.Enabled = True

cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Select Nomination Event, Target, and Platform
```

```
before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datFireCommand.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()

    If cmdUpdate.Caption = "&Update" And _
        datFireCommand.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboTarget.Enabled = True
        cboNomination.Enabled = True
        cboPlatform.Enabled = True
        txtTimeSent.Enabled = True
        txtTimeRcvd.Enabled = True
        txtOCCCId.Enabled = True
        chkEngage.Enabled = True
        cmdAdd.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        datFireCommand.Enabled = False
        datFireCommand.Recordset.Edit

    Else
        If datFireCommand.Recordset.RecordCount > 0 Then
            datFireCommand.Recordset.Update

            cboTarget.Enabled = False
            cboNomination.Enabled = False
            cboPlatform.Enabled = False
            txtTimeSent.Enabled = False
            txtTimeRcvd.Enabled = False
            txtOCCCId.Enabled = False
            chkEngage.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datFireCommand.Enabled = True


        End If
    End If
```

```
End Sub

Private Sub datFireCommand_Reposition()

    SetFireCommandRecordNumber

End Sub

Private Sub Form_Load()

    datFireCommand.DatabaseName = gstNewDatabase

    stSQL1 = "Select Nomination from Nomination"
    stSQL2 = "Select TargetId from Target"
    stSQL3 = "Select Platform from Platform"


    Set rsNomination = db.OpenRecordset(stSQL1)
    Set rsTarget = db.OpenRecordset(stSQL2)
    Set rsPlatform = db.OpenRecordset(stSQL3)

    'fill cboNomination
    Do Until rsNomination.EOF
        cboNomination.AddItem rsNomination!Nomination
        rsNomination.MoveNext
    Loop

    'fill cboTarget
    Do Until rsTarget.EOF
        cboTarget.AddItem rsTarget!TargetId
        rsTarget.MoveNext
    Loop

    'fill cboPlatform
    Do Until rsPlatform.EOF
        cboPlatform.AddItem rsPlatform!Platform
        rsPlatform.MoveNext
    Loop

    With datFireCommand
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetFireCommandRecordNumber

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
```

```
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub SetFireCommandRecordNumber()

    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datFireCommand.Recordset.RecordCount
    iCurrentRecord = datFireCommand.Recordset.AbsolutePosition + 1

    If datFireCommand.Recordset.EOF Then
        datFireCommand.Caption = "No more records"
    Else
        datFireCommand.Caption = "Fire Command Event Record " &
iCurrentRecord & _
                                 " of " & iRecordCount
    End If

End Sub

Private Sub mnuFilePrint_Click()

    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datFireCommand.Recordset

            If bWord = True Then

                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=6


                Sel.TypeText Text:="FireCommand"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Nomination"
                Sel.MoveRight unit:=12                  '12=next cell
```

```
                Sel.TypeText Text:="TargetId"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FCTimeXmit"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FCTimeRcvd"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FirerPlatform"
                Sel.MoveRight unit:=12                    '12=next cell

                Do Until .EOF

                    Sel.TypeText Text:=!FireCommand
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!Nomination
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!TargetId
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!FCTimeXmit
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!FCTimeRcvd
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!FirerPlatform
                    Sel.MoveRight unit:=12                    '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\FireCmds.txt" For Output As #1

                    Print #1, "FireCommand"; Chr(9); "Nomination";
Chr(9); "TargetId"; Chr(9); _
```

```
                                "FCTimeXmit"; Chr(9); "FCTimeRcvd";
Chr(9); _
                                "FirerPlatform"; Chr(9)
                    Do Until .EOF

                        Print #1, !FireCommand; Chr(9); _
                                  !Nomination; Chr(9); _
                                  !TargetId; Chr(9); _
                                  !FCTimeXmit; Chr(9); _
                                  !FCTimeRcvd; Chr(9); _
                                  !FirePlatform; Chr(9)

                        .MoveNext
                    Loop

                    Close #1

                End If
            End If

        .MoveFirst

        End With

    End If

    bContinue = False
    bWord = False
    bText = False


mnuPrintErrors:
        Select Case Err.Number
            Case 94
                Sel.TypeText Text:=""
                Resume Next
        End Select

End Sub

Private Sub txtNomination_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsNomination.MoveFirst
    If txtNomination <> "" Then
        Do Until iIndex = rsNomination.RecordCount Or bFound
            If rsNomination!Nomination = txtNomination Then
                cboNomination.Text = rsNomination!Nomination
                bFound = True
            Else
                rsNomination.MoveNext
                iIndex = iIndex + 1
```

```
            End If

        Loop
    End If

End Sub

Private Sub txtPlatform_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsPlatform.MoveFirst
    If txtPlatform <> "" Then
        Do Until iIndex = rsPlatform.RecordCount Or bFound
            If rsPlatform!Platform = txtPlatform Then
                cboPlatform.Text = rsPlatform!Platform
                bFound = True
            Else
                rsPlatform.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub

Private Sub txtTarget_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsTarget.MoveFirst
    If txtTarget <> "" Then
        Do Until iIndex = rsTarget.RecordCount Or bFound
            If rsTarget!TargetId = txtTarget Then
                cboTarget.Text = rsTarget!TargetId
                bFound = True
            Else
                rsTarget.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub


'*************************************************************
'Module:        frmFireEvent.frm
```

```
'Description:   Allows user to access the fire event records
'               for addition, deletion, and modification.
'Programmer:    Kevin Colón
'*************************************************************

Option Explicit

Dim rsFireCommand   As Recordset
Dim stSQL           As String
Private WordApp     As Word.Application
Private Doc         As Word.Document
Private Sel         As Word.Selection


Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add Event" Then

        datFire.Recordset.AddNew
        cboFireCommand.Enabled = True
        cboFireCommand.ListIndex = -1
        txtTime.Enabled = True
        txtWeaponMagStat.Enabled = True
        txtLocation.Enabled = True
        txtAltitude.Enabled = True
        txtRounds.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datFire.Enabled = False

    Else

        datFire.Recordset.CancelUpdate
        cboFireCommand.Enabled = False
        txtTime.Enabled = False
        txtWeaponMagStat.Enabled = False
        txtLocation.Enabled = False
        txtAltitude.Enabled = False
        txtRounds.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
        datFire.Enabled = True
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
```

```
    Exit Sub

HandleAddErrors:
    Dim stMess      As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0     'turn off error trapping

End Sub

Private Sub cmdDel_Click()

    Dim iResp       As Integer

    On Error GoTo HandleDelErrors

    If datFire.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Event " & txtFire & "?", vbYesNo, _
                    "Delete Event")
        If iResp = vbYes Then
            With datFire.Recordset
                .Delete
                .MoveNext
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation, "Delete Event"

    End If

cmdDel_Click:
    Exit Sub

HandleDelErrors:
    Dim stMess       As String

    stMess = "Cannot complete operation." & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0


End Sub

Private Sub cmdSave_Click()

    'save current record
```

```
    On Error GoTo HandleSaveErrors
    If cboFireCommand.ListIndex >= 0 And txtRounds <> "" Then
        If Val(txtCounter) < 10 Then
            txtFire.Text = "FE0000" & txtCounter.Text
        Else
            If Val(txtCounter) < 100 Then
                txtFire.Text = "FE000" & txtCounter.Text
            Else
                If Val(txtCounter) < 1000 Then
                    txtFire.Text = "FE00" & txtCounter.Text
                Else
                    If Val(txtCounter) < 10000 Then
                        txtFire.Text = "FE0" & txtCounter.Text
                    Else
                        txtFire.Text = "FE" & txtCounter.Text
                    End If
                End If
            End If
        End If

        datFire.Recordset.Update
    Else
        MsgBox "You must select an Fire Command Event and enter number
of rounds before saving." _
                , vbExclamation, "Add Fire Event"
        datFire.Recordset.CancelUpdate
    End If

    cboFireCommand.Enabled = False
    txtTime.Enabled = False
    txtWeaponMagStat.Enabled = False
    txtLocation.Enabled = False
    txtAltitude.Enabled = False
    txtRounds.Enabled = False
    cmdUpdate.Enabled = True
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    cmdAdd.Caption = "&Add Event"
    mnuFile.Enabled = True
    datFire.Enabled = True
    cmdAdd.SetFocus


    datFire.Enabled = True

cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping
```

```
        Case 3058, 3315      'no entry in key field
            stMess = "Select Fire Command Event and enter number of
rounds before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0      'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datFire.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()

    If cmdUpdate.Caption = "&Update" And _
        datFire.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboFireCommand.Enabled = True
        txtTime.Enabled = True
        txtWeaponMagStat.Enabled = True
        txtLocation.Enabled = True
        txtAltitude.Enabled = True
        txtRounds.Enabled = True
        cmdAdd.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        datFire.Enabled = False
        datFire.Recordset.Edit

    Else
        If datFire.Recordset.RecordCount > 0 Then
            datFire.Recordset.Update

            cboFireCommand.Enabled = False
            txtTime.Enabled = False
            txtWeaponMagStat.Enabled = False
            txtLocation.Enabled = False
            txtAltitude.Enabled = False
            txtRounds.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datFire.Enabled = True


        End If
```

```
    End If

End Sub

Private Sub datFire_Reposition()

    SetFireRecordNumber

End Sub

Private Sub Form_Load()

    datFire.DatabaseName = gstNewDatabase

    stSQL = "Select FireCommand from FireCommand"

    Set rsFireCommand = db.OpenRecordset(stSQL)

    'fill cboFireCommand
    Do Until rsFireCommand.EOF
        cboFireCommand.AddItem rsFireCommand!FireCommand
        rsFireCommand.MoveNext
    Loop


    With datFire
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetFireRecordNumber

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub SetFireRecordNumber()

    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer
```

```
    iRecordCount = datFire.Recordset.RecordCount
    iCurrentRecord = datFire.Recordset.AbsolutePosition + 1

    If datFire.Recordset.EOF Then
        datFire.Caption = "No more records"
    Else
        datFire.Caption = "Fire Event Record " & iCurrentRecord & _
                          " of " & iRecordCount
    End If

End Sub


Private Sub mnuFilePrint_Click()
    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datFire.Recordset

            If bWord = True Then

                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=6


                Sel.TypeText Text:="Fire"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FireCommand"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FireTime"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FirerLocation"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FirerAltitude"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="RoundsFired"
                Sel.MoveRight unit:=12                    '12=next cell

                Do Until .EOF

                    Sel.TypeText Text:=!Fire
```

```
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!FireCommand
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!FireTime
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!FirerLocation
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!FirerAltitude
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!RoundsFired
                    Sel.MoveRight unit:=12                  '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\FireEvents.txt" For Output As #1

                    Print #1, "Fire"; Chr(9); "FireCommand"; Chr(9);
"FireTime"; Chr(9); _
                                "FirerLocation"; Chr(9);
"FirerAltitude"; Chr(9); _
                                "RoundsFired"; Chr(9)

                    Do Until .EOF

                        Print #1, !Fire; Chr(9); _
                                  !FireCommand; Chr(9); _
                                  !FireTime; Chr(9); _
                                  !FirerLocation; Chr(9); _
                                  !FirerAltitude; Chr(9); _
                                  !RoundFired; Chr(9)

                        .MoveNext
                    Loop
```

```
                    Close #1

                End If
            End If

        .MoveFirst

        End With

    End If

    bContinue = False
    bWord = False
    bText = False


mnuPrintErrors:
        Select Case Err.Number
            Case 94
                Sel.TypeText Text:=""
                Resume Next
        End Select

End Sub



'*************************************************************
'Module:         frmGISRS.frm
'Description:    Allows user to access the GISRS terminal
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit
Dim rsPlatform      As Recordset
Dim stSQL           As String

Private Sub cboPlatform_Click()

    If cboPlatform.ListIndex >= 0 Then

        txtPlatform = cboPlatform.Text

    End If

End Sub

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datGISRS.Recordset.AddNew
        txtTerminal.Enabled = True
```

```
        txtTerminal.SetFocus
        txtFunction.Enabled = True
        cboPlatform.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datGISRS.Enabled = False

    Else
        datGISRS.Recordset.CancelUpdate
        txtTerminal.Enabled = False
        txtFunction.Enabled = False
        cboPlatform.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datGISRS.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datGISRS.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Terminal " & txtTerminal.Text & "?",
vbYesNo, "Delete Terminal")
        If iResp = vbYes Then
            With datGISRS.Recordset
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
```

```
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Terminal"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtTerminal.Text <> "" Then
        txtTerminal.Text = UCase(txtTerminal.Text)
        iResp = MsgBox("Do you want to add " & txtTerminal.Text & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Terminal")
        If iResp = vbYes Then
            datGISRS.Recordset.Update
        End If

    Else
        MsgBox "You must enter a Terminal before saving.",
vbExclamation _
                , "Add Terminal"
        datGISRS.Recordset.CancelUpdate
    End If

    txtTerminal.Enabled = False
    txtFunction.Enabled = False
    cboPlatform.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datGISRS.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True
```

```
cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a location before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datGISRS.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datGISRS.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtTerminal.Enabled = True
        txtFunction.Enabled = True
        cboPlatform.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtTerminal.SetFocus
        cmdAdd.Enabled = False
        datGISRS.Enabled = False
        datGISRS.Recordset.Edit
    Else
        If datGISRS.Recordset.RecordCount > 0 Then
            txtTerminal = UCase(txtTerminal)
            datGISRS.Recordset.Update

            txtTerminal.Enabled = False
            txtFunction.Enabled = False
            cboPlatform.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datGISRS.Enabled = True
```

```
        End If
    End If

End Sub

Private Sub datGISRS_Reposition()

    SetTerminalRecordNumber

End Sub

Private Sub FillPlatformCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
    cboPlatform.Clear

    With rsPlatform

        iCount = .RecordCount

        'fill the list
        Do Until .EOF
            If !Platform <> "" Then
                cboPlatform.AddItem !Platform

            End If
            .MoveNext
        Loop
    End With


End Sub

Private Sub Form_Load()

    datGISRS.DatabaseName = gstNewDatabase

    stSQL = "Select Platform from Platform"

    Set rsPlatform = db.OpenRecordset(stSQL)

    FillPlatformCombo

    With datGISRS
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetTerminalRecordNumber

End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datGISRS.Recordset.FindFirst "[GISRSTerminal] = '" & _
                InputBox("Enter the GISRS Terminal", "GISRS Terminal
Search") & "'"

    If datGISRS.Recordset.NoMatch Then
        MsgBox "GISRS Terminal was not found.", vbOKOnly, "GISRS
Terminal Search"
        datGISRS.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetTerminalRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datGISRS.Recordset.RecordCount
    iCurrentRecord = datGISRS.Recordset.AbsolutePosition + 1
    If datGISRS.Recordset.EOF Then
        datGISRS.Caption = "No more records"
    Else
        datGISRS.Caption = "Terminal " & iCurrentRecord & _
                           " of " & iRecordCount
    End If

End Sub

Private Sub txtPlatform_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
    Dim bFound  As Boolean

    rsPlatform.MoveFirst
    If txtPlatform <> "" Then
        Do Until iIndex = rsPlatform.RecordCount Or bFound
            If rsPlatform!Platform = txtPlatform Then
                cboPlatform.Text = rsPlatform!Platform
```

```
                bFound = True
            Else
                rsPlatform.MoveNext
                iIndex = iIndex + 1
            End If
        Loop
    Else
        cboPlatform.ListIndex = -1
    End If

End Sub


'************************************************************
'Module:         frmImpactEvent.frm
'Description:    Allows user to access the impact event
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Dim rsFire    As Recordset
Dim rsPlatform    As Recordset
Dim rsSensor          As Recordset
Dim stSQL1           As String
Dim stSQL2           As String
Dim stSQL3           As String
Private WordApp      As Word.Application
Private Doc          As Word.Document
Private Sel          As Word.Selection


Private Sub cboFire_Change()

    If cboFire.ListIndex >= 0 Then
        txtFire = cboFire.Text
    End If

End Sub

Private Sub cboSensor_Change()

    If cboSensor.ListIndex >= 0 Then
        txtSensor = cboSensor.Text
    End If

End Sub

Private Sub cboPlatform_Change()

    If cboPlatform.ListIndex >= 0 Then
        txtPlatform = cboPlatform.Text
    End If
```

```
End Sub

Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add Event" Then

        datImpact.Recordset.AddNew
        cboFire.Enabled = True
        cboPlatform.Enabled = True
        cboSensor.Enabled = True
        cboFire.ListIndex = -1
        cboPlatform.ListIndex = -1
        cboSensor.ListIndex = -1
        txtImpactTime.Enabled = True
        txtBDATime.Enabled = True
        txtBDA.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datImpact.Enabled = False

    Else

        datImpact.Recordset.CancelUpdate
        cboFire.Enabled = False
        cboPlatform.Enabled = False
        cboSensor.Enabled = False
        txtImpactTime.Enabled = False
        txtBDATime.Enabled = False
        txtBDA.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
        datImpact.Enabled = True
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess      As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0     'turn off error trapping
```

```
End Sub

Private Sub cmdDel_Click()

    Dim iResp       As Integer

    On Error GoTo HandleDelErrors

    If datImpact.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Event " & txtImpact & "?", vbYesNo, _
                    "Delete Event")
        If iResp = vbYes Then
            With datImpact.Recordset
                .Delete
                .MoveNext
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation, "Delete Event"

    End If

cmdDel_Click:
    Exit Sub

HandleDelErrors:
    Dim stMess       As String

    stMess = "Cannot complete operation." & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0


End Sub

Private Sub cmdSave_Click()

    'save current record
    On Error GoTo HandleSaveErrors
    If cboFire.ListIndex >= 0 Then
        If Val(txtCounter) < 10 Then
            txtImpact.Text = "IE0000" & txtCounter.Text
        Else
            If Val(txtCounter) < 100 Then
                txtImpact.Text = "IE000" & txtCounter.Text
            Else
                If Val(txtCounter) < 1000 Then
```

```
                    txtImpact.Text = "IE00" & txtCounter.Text
                Else
                    If Val(txtCounter) < 10000 Then
                        txtImpact.Text = "IE0" & txtCounter.Text
                    Else
                        txtImpact.Text = "IE" & txtCounter.Text
                    End If
                End If
            End If
        End If

        datImpact.Recordset.Update
    Else
        MsgBox "You must select a Fire Event before saving." _
                , vbExclamation, "Add Impact Event"
        datImpact.Recordset.CancelUpdate
    End If

    cboFire.Enabled = False
    cboPlatform.Enabled = False
    cboSensor.Enabled = False
    txtImpactTime.Enabled = False
    txtBDATime.Enabled = False
    txtBDA.Enabled = False
    cmdUpdate.Enabled = True
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    cmdAdd.Caption = "&Add Event"
    mnuFile.Enabled = True
    datImpact.Enabled = True
    cmdAdd.SetFocus


    datImpact.Enabled = True

cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Select Fire Event before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
```

```
            datImpact.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()

    If cmdUpdate.Caption = "&Update" And _
        datImpact.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboFire.Enabled = True
        cboPlatform.Enabled = True
        cboSensor.Enabled = True
        txtImpactTime.Enabled = True
        txtBDATime.Enabled = True
        txtBDA.Enabled = True
        cmdAdd.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        datImpact.Enabled = False
        datImpact.Recordset.Edit

    Else
        If datImpact.Recordset.RecordCount > 0 Then
            datImpact.Recordset.Update

            cboFire.Enabled = False
            cboPlatform.Enabled = False
            cboSensor.Enabled = False
            txtImpactTime.Enabled = False
            txtBDATime.Enabled = False
            txtBDA.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datImpact.Enabled = True


        End If
    End If

End Sub

Private Sub datImpact_Reposition()

    SetImpactRecordNumber

End Sub

Private Sub Form_Load()
```

```
    datImpact.DatabaseName = gstNewDatabase

    stSQL1 = "Select Fire from Fire"
    stSQL2 = "Select Platform from Platform"
    stSQL3 = "Select SensorType from SensorType"


    Set rsFire = db.OpenRecordset(stSQL1)
    Set rsPlatform = db.OpenRecordset(stSQL2)
    Set rsSensor = db.OpenRecordset(stSQL3)

    'fill cboFire
    Do Until rsFire.EOF
        cboFire.AddItem rsFire!Fire
        rsFire.MoveNext
    Loop

    'fill cboPlatform
    Do Until rsPlatform.EOF
        cboPlatform.AddItem rsPlatform!Platform
        rsPlatform.MoveNext
    Loop

    'fill cboSensor
    Do Until rsSensor.EOF
        cboSensor.AddItem rsSensor!SensorType
        rsSensor.MoveNext
    Loop

    With datImpact
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetImpactRecordNumber

End Sub

Private Sub SetImpactRecordNumber()

    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datImpact.Recordset.RecordCount
    iCurrentRecord = datImpact.Recordset.AbsolutePosition + 1

    If datImpact.Recordset.EOF Then
        datImpact.Caption = "No more records"
    Else
        datImpact.Caption = "Impact Event Record " & iCurrentRecord & _
                            " of " & iRecordCount
```

```
    End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFilePrint_Click()
    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datImpact.Recordset

            If bWord = True Then

                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=7


                Sel.TypeText Text:="Impact"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="FireEvent"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="ImpactTime"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="BDA"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="BDATime"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Platform"
                Sel.MoveRight unit:=12                    '12=next cell
```

```
                Sel.TypeText Text:="SensorType"
                Sel.MoveRight unit:=12                    '12=next cell

                Do Until .EOF

                    Sel.TypeText Text:=!Impact
                    Sel.MoveRight unit:=12                '12=next
cell

                    Sel.TypeText Text:=!FireEvent
                    Sel.MoveRight unit:=12                '12=next
cell

                    Sel.TypeText Text:=!ImpactTime
                    Sel.MoveRight unit:=12                '12=next
cell

                    Sel.TypeText Text:=!BDA
                    Sel.MoveRight unit:=12                '12=next
cell

                    Sel.TypeText Text:=!BDATime
                    Sel.MoveRight unit:=12                '12=next
cell

                    Sel.TypeText Text:=!Platform
                    Sel.MoveRight unit:=12                '12=next
cell

                    Sel.TypeText Text:=!SensorType
                    Sel.MoveRight unit:=12                '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\ImpactEvents.txt" For Output As
#1

                    Print #1, "Impact"; Chr(9); "FireEvent"; Chr(9);
"ImpactTime"; Chr(9); _
                                "BDA"; Chr(9); "BDATime"; Chr(9); _
                                "Platform"; Chr(9); "SensorType";
Chr(9)
```

```
                    Do Until .EOF

                        Print #1, !Impact; Chr(9); _
                                  !FireEvent; Chr(9); _
                                  !ImpactTime; Chr(9); _
                                  !BDA; Chr(9); _
                                  !BDATime; Chr(9); _
                                  !Platform; Chr(9); _
                                  !SensorType; Chr(9)

                        .MoveNext
                    Loop

                    Close #1

                End If
            End If

        .MoveFirst

        End With

    End If

    bContinue = False
    bWord = False
    bText = False


mnuPrintErrors:
        Select Case Err.Number
            Case 94
                Sel.TypeText Text:=""
                Resume Next
        End Select

End Sub

Private Sub txtFire_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsFire.MoveFirst
    If txtFire <> "" Then
        Do Until iIndex = rsFire.RecordCount Or bFound
            If rsFire!Fire = txtFire Then
                cboFire.Text = rsFire!Fire
                bFound = True
            Else
                rsFire.MoveNext
                iIndex = iIndex + 1
            End If
```

```
        Loop
    End If

End Sub

Private Sub txtSensor_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsSensor.MoveFirst
    If txtSensor <> "" Then
        Do Until iIndex = rsSensor.RecordCount Or bFound
            If rsSensor!SensorType = txtSensor Then
                cboSensor.Text = rsSensor!SensorType
                bFound = True
            Else
                rsSensor.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub

Private Sub txtPlatform_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsPlatform.MoveFirst
    If txtPlatform <> "" Then
        Do Until iIndex = rsPlatform.RecordCount Or bFound
            If rsPlatform!Platform = txtPlatform Then
                cboPlatform.Text = rsPlatform!Platform
                bFound = True
            Else
                rsPlatform.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub


'************************************************************
'Module:        frmInitiatives.frm
'Description:   Allows user to access the initiatives
'               records for addition, deletion, and
'               modification.
```

```
'Programmer:    Kevin Colón
'*************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datInitiatives.Recordset.AddNew
        txtInitiative.Enabled = True
        txtInitiative.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datInitiatives.Enabled = False

    Else
        datInitiatives.Recordset.CancelUpdate
        txtInitiative.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datInitiatives.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
           & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping


End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datInitiatives.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Initiative " & txtInitiative.Text & "?",
```

```
vbYesNo, "Delete Initiative")
        If iResp = vbYes Then
            With datInitiatives.Recordset
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Initiative"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtInitiative <> "" And txtDescription <> "" Then
        txtInitiative = UCase(txtInitiative)
        iResp = MsgBox("Do you want to add " & txtInitiative & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Initiative")
        If iResp = vbYes Then
            datInitiatives.Recordset.Update
        End If

    Else
        MsgBox "You must enter an Initiative and a description before
saving.", vbExclamation _
                , "Add Initiative"
        datInitiatives.Recordset.CancelUpdate
    End If

    txtInitiative.Enabled = False
```

```
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datInitiatives.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a Initiative name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datInitiatives.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datInitiatives.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtInitiative.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtInitiative.SetFocus
        cmdAdd.Enabled = False
        datInitiatives.Enabled = False
        datInitiatives.Recordset.Edit
    Else
        If datInitiatives.Recordset.RecordCount > 0 Then
            datInitiatives.Recordset.Update

            txtInitiative.Enabled = False
            txtDescription.Enabled = False
```

```
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datInitiatives.Enabled = True
        End If
    End If

End Sub

Private Sub datInitiatives_Reposition()
    SetInitiativeRecordNumber

End Sub

Private Sub Form_Load()
    datInitiatives.DatabaseName = gstNewDatabase

    With datInitiatives
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetInitiativeRecordNumber
'

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datInitiatives.Recordset.FindFirst "[Description] = '" & _
                InputBox("Enter the Initiative", "Initiative Search") &
"'"

    If datInitiatives.Recordset.NoMatch Then
        MsgBox "Initiative was not found.", vbOKOnly, "Initiative
Search"
```

```
        datInitiatives.Recordset.MoveFirst          'go to first record
    End If

End Sub

Private Sub SetInitiativeRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datInitiatives.Recordset.RecordCount
    iCurrentRecord = datInitiatives.Recordset.AbsolutePosition + 1
    If datInitiatives.Recordset.EOF Then
        datInitiatives.Caption = "No more records"
    Else
        datInitiatives.Caption = "Initiative " & iCurrentRecord & _
                    " of " & iRecordCount
    End If

End Sub


'**************************************************************
'Module:        frmLAWS.frm
'Description:   Allows user to access the LAWS terminal
'               records for addition, deletion, and
'               modification.
'Programmer:    Kevin Colón
'**************************************************************

Option Explicit
Dim rsPlatform      As Recordset
Dim stSQL           As String

Private Sub cboPlatform_Click()

    If cboPlatform.ListIndex >= 0 Then

        txtPlatform = cboPlatform.Text

    End If

End Sub

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datLAWS.Recordset.AddNew
        txtTerminal.Enabled = True
        txtTerminal.SetFocus
        txtFunction.Enabled = True
        cboPlatform.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
```

```
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datLAWS.Enabled = False

    Else
        datLAWS.Recordset.CancelUpdate
        txtTerminal.Enabled = False
        txtFunction.Enabled = False
        cboPlatform.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datLAWS.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
          & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datLAWS.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Terminal " & txtTerminal.Text & "?",
vbYesNo, "Delete Terminal")
        If iResp = vbYes Then
            With datLAWS.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
```

```
                , "Delete Terminal"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtTerminal.Text <> "" Then
        txtTerminal.Text = UCase(txtTerminal.Text)
        iResp = MsgBox("Do you want to add " & txtTerminal.Text & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Terminal")
        If iResp = vbYes Then
            datLAWS.Recordset.Update
        End If

    Else
        MsgBox "You must enter a Terminal before saving.",
vbExclamation _
                , "Add Terminal"
        datLAWS.Recordset.CancelUpdate
    End If

    txtTerminal.Enabled = False
    txtFunction.Enabled = False
    cboPlatform.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datLAWS.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
```

```
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a location before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datLAWS.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datLAWS.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtTerminal.Enabled = True
        txtFunction.Enabled = True
        cboPlatform.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtTerminal.SetFocus
        cmdAdd.Enabled = False
        datLAWS.Enabled = False
        datLAWS.Recordset.Edit
    Else
        If datLAWS.Recordset.RecordCount > 0 Then
            txtTerminal = UCase(txtTerminal)
            datLAWS.Recordset.Update

            txtTerminal.Enabled = False
            txtFunction.Enabled = False
            cboPlatform.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datLAWS.Enabled = True
        End If
    End If

End Sub

Private Sub datLAWS_Reposition()
```

```
    SetTerminalRecordNumber

End Sub

Private Sub FillPlatformCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
    cboPlatform.Clear

    With rsPlatform
        iCount = .RecordCount

        'fill the list
        Do Until .EOF
            If !Platform <> "" Then
                cboPlatform.AddItem !Platform

            End If
            .MoveNext
        Loop
    End With


End Sub

Private Sub Form_Load()

    datLAWS.DatabaseName = gstNewDatabase

    stSQL = "Select Platform from Platform"
    Set rsPlatform = db.OpenRecordset(stSQL)

    FillPlatformCombo

    With datLAWS
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetTerminalRecordNumber

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub
```

```
Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datLAWS.Recordset.FindFirst "[LAWSTerminal] = '" & _
                InputBox("Enter the LAWS Terminal", "LAWS Terminal
Search") & "'"

    If datLAWS.Recordset.NoMatch Then
        MsgBox "LAWS Terminal was not found.", vbOKOnly, "LAWS Terminal
Search"
        datLAWS.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetTerminalRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datLAWS.Recordset.RecordCount
    iCurrentRecord = datLAWS.Recordset.AbsolutePosition + 1
    If datLAWS.Recordset.EOF Then
        datLAWS.Caption = "No more records"
    Else
        datLAWS.Caption = "Terminal " & iCurrentRecord & _
                          " of " & iRecordCount
    End If

End Sub

Private Sub txtPlatform_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
    Dim bFound  As Boolean

    rsPlatform.MoveFirst

    If txtPlatform <> "" Then
        Do Until iIndex = rsPlatform.RecordCount Or bFound
            If rsPlatform!Platform = txtPlatform Then
                cboPlatform.Text = rsPlatform!Platform
                bFound = True
            Else
                rsPlatform.MoveNext
                iIndex = iIndex + 1
            End If
        Loop
```

```
    Else

        cboPlatform.ListIndex = -1

    End If


End Sub


'*************************************************************
'Module:         frmMain.frm
'Description:    Allows user to access the forms linked to
'                the database by use of dropdown menus.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

    Dim rsAcquisition As Recordset
    Dim rsFire As Recordset
    Dim rsFireCommand As Recordset
    Dim rsImpact As Recordset
    Dim rsMensuration As Recordset
    Dim rsNomination As Recordset
    Dim rsTarget As Recordset
    Dim rsLAWSdata As Recordset
    Dim rsGISRS As Recordset
    Dim rsLAWS As Recordset
    Dim rsPlatform As Recordset
    Dim rsSensor As Recordset
    Dim rsThreat As Recordset
    Dim rsWeapon As Recordset
    Dim stSQL1 As String
    Dim stSQL2 As String
    Dim stSQL3 As String
    Dim stSQL4 As String
    Dim stSQL5 As String
    Dim stSQL6 As String
    Dim stSQL7 As String
    Dim stSQL8 As String
    Dim stSQL9 As String
    Dim stSQL10 As String
    Dim stSQL11 As String
    Dim stSQL12 As String
    Dim stSQL13 As String
    Dim stSQL14 As String
    Dim stSearch As String
    Dim stMessage As String
    Dim bGFound As Boolean
    Dim bLFound As Boolean


Private Sub AddAcquisition()
```

```
    rsAcquisition.AddNew

    If rsAcquisition!AcquisitionCounter < 10 Then

        rsAcquisition!Acquisition = "AE0000" &
rsAcquisition!AcquisitionCounter

    Else
        If rsAcquisition!AcquisitionCounter < 100 Then

            rsAcquisition!Acquisition = "AE000" &
rsAcquisition!AcquisitionCounter

        Else
            If rsAcquisition!AcquisitionCounter < 1000 Then

                rsAcquisition!Acquisition = "AE00" &
rsAcquisition!AcquisitionCounter

            Else
                If rsAcquisition!AcquisitionCounter < 10000 Then

                    rsAcquisition!Acquisition = "AE0" &
rsAcquisition!AcquisitionCounter

                Else
                    rsAcquisition!Acquisition = "AE" &
rsAcquisition!AcquisitionCounter

                End If
            End If
        End If
    End If

    rsAcquisition!TrackId = rsLAWSdata!TrackId
    rsAcquisition!ThreatType = rsLAWSdata!ThreatType

    'get string from LAWS data and use it for platform search to
    'fill platform field in Acquisition table from LAWSdata info
    If rsLAWSdata!Nominator <> "" Then

        stSearch = rsLAWSdata!Nominator

        rsGISRS.MoveFirst
        rsLAWS.MoveFirst
        bGFound = False
        bLFound = False

        'checks GISRS table
        Do Until rsGISRS.EOF Or bGFound = True
            If rsGISRS!GISRSTerminal = stSearch Then

                bGFound = True
```

```
                rsAcquisition!AcqPlatform = rsGISRS!Location

            Else
                rsGISRS.MoveNext

            End If
        Loop

        'checks LAWS table if search not successful in GISRS table
        If bGFound = False Then

            Do Until rsLAWS.EOF Or bLFound = True

                If rsLAWS!LAWSTerminal = stSearch Then

                    bLFound = True
                    rsAcquisition!AcqPlatform = rsLAWS!Location

                Else
                    rsLAWS.MoveNext

                End If
            Loop
        End If
    End If

    rsAcquisition!TrackLocation = rsLAWSdata!TargetLocation
    rsAcquisition!TrackAltitude = rsLAWSdata!Altitude
    rsAcquisition!Remark = rsLAWSdata!ThreatDescription


    rsAcquisition!AcqSensorType = rsLAWSdata!AcqSensor

    If rsLAWSdata!AcqSensor = Null Then

        rsAcquisition!AcqSensorType = "None"
    Else
        rsAcquisition!AcqSensorType = rsLAWSdata!AcqSensor

    End If

    If rsLAWSdata!AcqTime <> Null Or rsLAWSdata!AcqTime <> "" Then

        rsAcquisition!AcqTime = rsLAWSdata!AcqTime

    End If

    rsAcquisition.Update


End Sub

Private Sub AddFire()
```

```
    rsFireCommand.MoveLast

    If rsLAWSdata!RoundsFired <> "" And rsLAWSdata!RoundsFired > 0 Then

        rsFire.AddNew

        If rsFire!FireCounter < 10 Then

            rsFire!Fire = "FE0000" & rsFire!FireCounter

        Else
            If rsFire!FireCounter < 100 Then

                rsFire!Fire = "FE000" & rsFire!FireCounter

            Else
                If rsFire!FireCounter < 1000 Then

                    rsFire!Fire = "FE00" & rsFire!FireCounter

                Else
                    If rsFire!FireCounter < 10000 Then

                        rsFire!Fire = "FE0" & rsFire!FireCounter

                    Else
                        rsFire!Fire = "FE" & rsFire!FireCounter

                    End If
                End If
            End If
        End If

        rsFire!FireCommand = rsFireCommand!FireCommand
        rsFire!RoundsFired = rsLAWSdata!RoundsFired
        rsFire!FireTime = rsLAWSdata!FireEventTime

        rsFire.Update

        AddImpact

    End If


End Sub

Private Sub AddFireCommand()

    Dim bPFound As Boolean

    rsTarget.MoveLast

    rsFireCommand.AddNew
```

```
    If rsFireCommand!FireCommandCounter < 10 Then

        rsFireCommand!FireCommand = "FC0000" &
rsFireCommand!FireCommandCounter

    Else
        If rsFireCommand!FireCommandCounter < 100 Then

            rsFireCommand!FireCommand = "FC000" &
rsFireCommand!FireCommandCounter

        Else
            If rsFireCommand!FireCommandCounter < 1000 Then

                rsFireCommand!FireCommand = "FC00" &
rsFireCommand!FireCommandCounter

            Else
                If rsFireCommand!FireCommandCounter < 10000 Then

                    rsFireCommand!FireCommand = "FC0" &
rsFireCommand!FireCommandCounter

                Else
                    rsFireCommand!FireCommand = "FC" &
rsFireCommand!FireCommandCounter

                End If
            End If
        End If
    End If

    rsFireCommand!Nomination = rsTarget!Nomination
    rsFireCommand!OCCCId = rsLAWSdata!TargetControl
    rsFireCommand!TargetId = rsTarget!TargetId

    stSearch = rsLAWSdata!FirerPlatform

    rsPlatform.MoveFirst

    Do Until rsPlatform.EOF Or bPFound = True
        If rsPlatform!LAWSFormat = stSearch Then

            bPFound = True
            rsFireCommand!FirerPlatform = rsPlatform!Platform

        Else
            rsPlatform.MoveNext

        End If

    Loop

    If bPFound = False Then
        rsFireCommand!FirerPlatform = rsLAWSdata!FirerPlatform & "-TAC"
```

```
    End If

    If rsLAWSdata!RoundsFired > 0 Then

        rsFireCommand!Engage = True

    End If

    rsFireCommand.Update

End Sub

Private Sub AddImpact()

    rsFire.MoveLast

    rsImpact.AddNew

    If rsImpact!ImpactCounter < 10 Then

        rsImpact!Impact = "IE0000" & rsImpact!ImpactCounter

    Else
        If rsImpact!ImpactCounter < 100 Then

            rsImpact!Impact = "IE000" & rsImpact!ImpactCounter

        Else
            If rsImpact!ImpactCounter < 1000 Then

                rsImpact!Impact = "IE00" & rsImpact!ImpactCounter

            Else
                If rsImpact!ImpactCounter < 10000 Then

                    rsImpact!Impact = "IE0" & rsImpact!ImpactCounter

                Else
                    rsImpact!Impact = "IE" & rsImpact!ImpactCounter

                End If
            End If
        End If
    End If

    rsImpact!FireEvent = rsFire!Fire
    rsImpact!ImpactTime = rsLAWSdata!ImpactTime

    rsImpact.Update

End Sub
```

```
Private Sub AddMensuration()

    rsAcquisition.MoveLast

    rsMensuration.AddNew

    If rsMensuration!MensurationCounter < 10 Then

        rsMensuration!Mensuration = "ME0000" &
rsMensuration!MensurationCounter

    Else
        If rsMensuration!MensurationCounter < 100 Then

            rsMensuration!Mensuration = "ME000" &
rsMensuration!MensurationCounter

        Else
            If rsMensuration!MensurationCounter < 1000 Then

                rsMensuration!Mensuration = "ME00" &
rsMensuration!MensurationCounter

            Else
                If rsMensuration!MensurationCounter < 10000 Then

                    rsMensuration!Mensuration = "ME0" &
rsMensuration!MensurationCounter

                Else
                    rsMensuration!Mensuration = "ME" &
rsMensuration!MensurationCounter

                End If
            End If
        End If
    End If

    rsMensuration!Acquisition = rsAcquisition!Acquisition
'     rsMensuration!MenSensorType = ?
'     rsMensuration!MenPlatform = ?

    stSearch = rsLAWSdata!Nominator

    rsGISRS.MoveFirst
    rsLAWS.MoveFirst
    bGFound = False
    bLFound = False

    'checks GISRS table
    Do Until rsGISRS.EOF Or bGFound = True
        If rsGISRS!GISRSTerminal = stSearch Then

            bGFound = True
```

```
            rsMensuration!GISRSTerminal = rsGISRS!GISRSTerminal

        Else
            rsGISRS.MoveNext

        End If
    Loop

    'checks LAWS table if search not successful in GISRS table
    If bGFound = False Then

        Do Until rsLAWS.EOF Or bLFound = True

            If rsLAWS!LAWSTerminal = stSearch Then

                bLFound = True
                rsMensuration!GISRSTerminal = rsLAWS!LAWSTerminal

            Else
                rsLAWS.MoveNext

            End If
        Loop
    End If

    rsMensuration.Update


End Sub

Private Sub AddNomination()

    rsMensuration.MoveLast

    rsNomination.AddNew

    If rsNomination!NominationCounter < 10 Then

        rsNomination!Nomination = "NE0000" &
rsNomination!NominationCounter

    Else
        If rsNomination!NominationCounter < 100 Then

            rsNomination!Nomination = "NE000" &
rsNomination!NominationCounter

        Else
            If rsNomination!NominationCounter < 1000 Then

                rsNomination!Nomination = "NE00" &
rsNomination!NominationCounter

            Else
                If rsNomination!NominationCounter < 10000 Then
```

```
                    rsNomination!Nomination = "NE0" &
rsNomination!NominationCounter

                Else
                    rsNomination!Nomination = "NE" &
rsNomination!NominationCounter

                End If
            End If
        End If
    End If

    rsNomination!Acquisition = rsMensuration!Acquisition
    rsNomination!Mensuration = rsMensuration!Mensuration

    stSearch = rsLAWSdata!Nominator

    rsGISRS.MoveFirst
    rsLAWS.MoveFirst

    bGFound = False
    bLFound = False

    'checks GISRS table
    Do Until rsGISRS.EOF Or bGFound = True
        If rsGISRS!GISRSTerminal = stSearch Then

            bGFound = True
            rsNomination!GISRSTerminal = rsGISRS!GISRSTerminal

        Else
            rsGISRS.MoveNext

        End If
    Loop

    'checks LAWS table if search not successful in GISRS table
    If bGFound = False Then

        Do Until rsLAWS.EOF Or bLFound = True

            If rsLAWS!LAWSTerminal = stSearch Then

                bLFound = True
                rsNomination!GISRSTerminal = rsLAWS!LAWSTerminal

            Else
                rsLAWS.MoveNext

            End If
        Loop
    End If

    rsNomination!TargetLocationError = rsLAWSdata!TLE
```

```
    rsNomination.Update


End Sub

Private Sub AddTarget()

    On Error GoTo HandleTargetError

    rsNomination.MoveLast

    rsTarget.AddNew
    rsTarget!TargetId = rsLAWSdata!TargetId
    rsTarget!TargetNLTTime = rsLAWSdata!NLTTime
    rsTarget!WeaponType = rsLAWSdata!WeaponType
    rsTarget!TargetLocation = rsLAWSdata!TargetLocation2
    rsTarget!Description = rsLAWSdata!TargetType
    rsTarget!Remark = rsLAWSdata!Remark
    rsTarget!Nomination = rsNomination!Nomination

    rsTarget.Update

AddTarget_Exit:

    Exit Sub

HandleTargetError:

    Select Case Err.Number
        Case 3022
            stMessage = "TargetId repeated.  Target " &
rsLAWSdata!TargetId _
                      & vbCrLf & " not saved as new record."
            MsgBox stMessage, vbOKOnly + vbInformation
            rsTarget.CancelUpdate
    End Select
    Resume


End Sub

Private Sub mnuDataSort_Click()

    Dim rsSorted        As Recordset
    Dim stPrevThreat    As String
    Dim stPrevTrack     As String
    Dim stPrevNominator As String
    Dim stNowThreat     As String
    Dim stNowTrack      As String
    Dim stNowNominator  As String
    Dim stNextThreat    As String
    Dim stNextTrack     As String
    Dim stNextNominator As String
    Dim stSQL1          As String
```

```
Dim stSQL2          As String

stSQL1 = "Select * from LAWSSorted"

Set rsSorted = db.OpenRecordset(stSQL1)

With rsSorted
    Do Until .EOF
        stNowThreat = !ThreatDescription

        If !TrackId <> "" Or !TrackId <> Null Then
            stNowTrack = !TrackId
        Else
            stNowTrack = ""
        End If

        If !Nominator <> "" Or !Nominator <> Null Then
            stNowNominator = !Nominator
        Else
            stNowNominator = ""
        End If

        If stNowTrack = "" Or stNowNominator = "" Then

            .MovePrevious

            If Not .BOF Then

                stPrevThreat = !ThreatDescription
                stPrevTrack = !TrackId
                stPrevNominator = !Nominator

                If stPrevThreat = stNowThreat Then
                    .MoveNext
                    .Edit
                    !TrackId = stPrevTrack
                    !Nominator = stPrevNominator
                    .Update

                Else
                    .MoveNext
                    .MoveNext
                    stNextThreat = !ThreatDescription
                    stNextTrack = !TrackId
                    stNextNominator = !Nominator
                    If stNextThreat = stNowThreat Then
                        .MovePrevious
                        .Edit
                        !TrackId = stNextTrack
                        !Nominator = stNextNominator
                        .Update

                    End If
```

```
                End If

            Else
                .MoveNext
                .MoveNext
                stNextThreat = !ThreatDescription
                stNextTrack = !TrackId
                stNextNominator = !Nominator
                .MovePrevious
                If stNextThreat = stNowThreat Then
                    .Edit
                    !TrackId = stNextTrack
                    !Nominator = stNextNominator
                    .Update
                End If


            End If
        End If

        .MoveNext

    Loop

End With


End Sub

Private Sub mnuDataTransfer_Click()
    Dim rsLAWSInfo  As Recordset
    Dim rsSorted    As Recordset
    Dim stLAWSInfo  As String
    Dim stSorted    As String
    Dim stTLE       As String

    stLAWSInfo = "Select * from LAWS"
    stSorted = "Select * from LAWSSorted"

    Set rsLAWSInfo = db.OpenRecordset(stLAWSInfo)
    Set rsSorted = db.OpenRecordset(stSorted)

    With rsLAWSInfo
        Do Until .EOF
            rsSorted.AddNew
            rsSorted!TargetId = !TargetId
            rsSorted!ThreatDescription = !ThreatDescription
            rsSorted!TargetLocation = !TargetLocation
            rsSorted!NLTTime = !NLTTime
            rsSorted!Altitude = !Altitude
            rsSorted!WeaponType = !WeaponType
            rsSorted!PlatLocation = !PlatLocation
            rsSorted!ThreatType = !ThreatType
            If !Remark1 <> "" Then
                stTLE = Left(rsLAWSInfo!Remark1, 2)
```

```
            End If

            If stTLE = "CE" Then
                rsSorted!TLE = !Remark1
            Else
                rsSorted!Remark = !Remark1

            End If

            rsSorted!AcqTime = !AcqTime
            rsSorted!AcqSensor = !AcqSensor
            rsSorted!TargetLocation2 = !TargetLocation2
            rsSorted!TargetType = !TargetType
            rsSorted!RoundsFired = !RoundsFired
            rsSorted!FirerPlatform = !FirerPlatform
            rsSorted!TargetControl = !TargetControl
            rsSorted!Priority = !Priority
            rsSorted!ImpactTime = !ImpactTime
            rsSorted!Nominator = !Nominator

            If !Remark2 <> "" Then
                stTLE = Left(rsLAWSInfo!Remark2, 2)
            End If

            If stTLE = "SH" Then
                rsSorted!Remark = !Remark2
            Else
                rsSorted!TrackId = !Remark2

            End If

            rsSorted!FireEventTime = !FireEventTime

            rsSorted.Update
            rsLAWSInfo.MoveNext

        Loop

    End With


End Sub

Private Sub mnuFileExit_Click()

    'terminates application
    End

End Sub

Private Sub mnuFileOpen_Click()
    'Select a different database (FBE)

    On Error GoTo HandleError
```

```
    With frmMain.dlgDatabase
        .FileName = gstNewDatabase
        .Filter = "Database files (*.mdb)|*.mdb|All files (*.*)|*.*"

        'if error encountered, skip next command
        On Error Resume Next
        .ShowOpen
        If Err.Number = cdlCancel Then
            gstNewDatabase = ""
        Else

            'set return filename to selected file
            gstNewDatabase = .FileName
            frmMain.Caption = .FileTitle & " Database"
        End If
    End With

    Set db = OpenDatabase(gstNewDatabase)

    'display Main form
    frmMain.Show

Sub_Exit:
    Exit Sub

HandleError:

    Select Case Err.Number
        Case 3004, 3024, 3044

            If gstNewDatabase = "" Then
                MsgBox "No database was selected.", vbExclamation,
"Database Error"

                'disables options only available when a database is
selected
                Me.mnuFileQueries.Enabled = False
                Me.mnuFileSQL.Enabled = False
                Me.mnuUpdate.Enabled = False

            Else
                Set db = OpenDatabase(gstNewDatabase)  'new database
location

                'reenables options once a database is selected
                Me.mnuFileQueries.Enabled = True
                Me.mnuFileSQL.Enabled = True
                Me.mnuUpdate.Enabled = True

                Resume      'open the database

            End If

        Case Else
```

```
            MsgBox Err.Description, vbOKOnly + vbExclamation,
"Unexpected Error"

            End             'exit the project

    End Select

End Sub

Private Sub mnuFileQueries_Click()
    frmQueries.Show
    Me.Enabled = False

End Sub

Private Sub mnuFileSQL_Click()

    frmSQL.Show
    Me.Enabled = False

End Sub

Private Sub mnuHelpAbout_Click()

    frmAbout.Show

End Sub

Private Sub mnuPopulate_Click()

    stSQL1 = "Select * from Acquisition"
    stSQL2 = "Select * from Fire"
    stSQL3 = "Select * from FireCommand"
    stSQL4 = "Select * from Impact"
    stSQL5 = "Select * from Mensuration"
    stSQL6 = "Select * from Nomination"
    stSQL7 = "Select * from Target"
    stSQL8 = "Select * from LawsSorted2"
    stSQL9 = "Select * from GISRSTerminal"
    stSQL10 = "Select * from LAWSTerminal"
    stSQL11 = "Select * from Platform"
    stSQL12 = "Select * from SensorType"
    stSQL13 = "Select * from ThreatType"
    stSQL14 = "Select * from WeaponType"

    Set rsAcquisition = db.OpenRecordset(stSQL1)
    Set rsFire = db.OpenRecordset(stSQL2)
    Set rsFireCommand = db.OpenRecordset(stSQL3)
    Set rsImpact = db.OpenRecordset(stSQL4)
    Set rsMensuration = db.OpenRecordset(stSQL5)
    Set rsNomination = db.OpenRecordset(stSQL6)
    Set rsTarget = db.OpenRecordset(stSQL7)
    Set rsLAWSdata = db.OpenRecordset(stSQL8)
    Set rsGISRS = db.OpenRecordset(stSQL9)
    Set rsLAWS = db.OpenRecordset(stSQL10)
```

```
    Set rsPlatform = db.OpenRecordset(stSQL11)
    Set rsSensor = db.OpenRecordset(stSQL12)
    Set rsThreat = db.OpenRecordset(stSQL13)
    Set rsWeapon = db.OpenRecordset(stSQL14)


    If rsLAWSdata.RecordCount > 0 Then

        Do Until rsLAWSdata.EOF       'fix to .EOF

            If rsAcquisition.RecordCount = 0 Then

                AddAcquisition
                AddMensuration
                AddNomination
                AddTarget
                AddFireCommand
                AddFire
                rsLAWSdata.MoveNext

            Else

                rsAcquisition.MoveLast

                If rsAcquisition!TrackId <> rsLAWSdata!TrackId Then

                    AddAcquisition
                    AddMensuration
                    AddNomination
                    AddTarget
                    AddFireCommand
                    AddFire
                    rsLAWSdata.MoveNext

                Else

                    AddTarget
                    AddFireCommand
                    AddFire
                    rsLAWSdata.MoveNext

                End If

            End If

        Loop

    Else

        stMessage = "No data to import from table: LawsSorted " &
vbCrLf _
            & "                    in database: " &
Me.dlgDatabase.FileTitle
        MsgBox stMessage, vbOKOnly + vbExclamation, "Data Population"
```

```
    End If

End Sub

Private Sub mnuUpdateAcquisition_Click()

    frmAcqEvents.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateAcronyms_Click()

    frmAcronyms.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateDataTypes_Click()

    frmDataTypes.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateFBE_Click()

    frmFBE.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateFire_Click()

    frmFireEvent.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateFireCommand_Click()

    frmFireCmdEvent.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateGISRS_Click()

    frmGISRS.Show
    Me.Enabled = False

End Sub
```

```
Private Sub mnuUpdateImpact_Click()

    frmImpactEvent.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateInitiatives_Click()

    frmInitiatives.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateLAWS_Click()

    frmLAWS.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateMensuration_Click()

    frmMenEvents.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateNomination_Click()

    frmNomEvents.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateObjectives_Click()

    frmObjectives.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdatePlatforms_Click()

    frmPlatforms.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdatePlatTypes_Click()

    frmPlatformTypes.Show
    Me.Enabled = False

End Sub
```

```
Private Sub mnuUpdateQuestions_Click()

    frmQuestions.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateSensTypes_Click()

    frmSensorTypes.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateTargets_Click()

    frmTargetEvents.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateThreatTypes_Click()

    frmThreatTypes.Show
    Me.Enabled = False

End Sub

Private Sub mnuUpdateWeaponTypes_Click()

    frmWeaponTypes.Show
    Me.Enabled = False

End Sub

Private Sub mnuViewTargets_Click()

    frmTargets2.Show
    Me.Enabled = False

End Sub


'*************************************************************
'Module:        frmMenEvents.frm
'Description:   Allows user to access the mensuration event
'               records for addition, deletion, and
'               modification.
'Programmer:    Kevin Colón
'*************************************************************

Option Explicit
```

```
Dim rsAcquisition   As Recordset
Dim rsSensor        As Recordset
Dim rsPlatform      As Recordset
Dim rsGISRS         As Recordset
Dim rsPTW           As Recordset
Dim stSQL1          As String
Dim stSQL2          As String
Dim stSQL3          As String
Dim stSQL4          As String
Dim stSQL5          As String
Private WordApp     As Word.Application
Private Doc         As Word.Document
Private Sel         As Word.Selection


Private Sub cboAcquisition_Click()

    If cboAcquisition.ListIndex >= 0 Then
        txtAcquisition = cboAcquisition.Text
    End If

End Sub

Private Sub cboGISRS_Click()

    If cboGISRS.ListIndex >= 0 Then
        txtGISRS = cboGISRS.Text
    End If

End Sub

Private Sub cboPlatform_Click()

    If cboPlatform.ListIndex >= 0 Then
        txtPlatform = cboPlatform.Text
    End If

End Sub

Private Sub cboPTW_Click()

    If cboPTW.ListIndex >= 0 Then
        txtPTW = cboPTW.Text
    End If

End Sub

Private Sub cboSensor_Click()

    If cboSensor.ListIndex >= 0 Then
        txtSensor = cboSensor.Text
    End If

End Sub
```

```
Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add Event" Then

        datMenEvents.Recordset.AddNew
        cboAcquisition.Enabled = True
        cboPlatform.Enabled = True
        cboSensor.Enabled = True
        cboGISRS.Enabled = True
        cboPTW.Enabled = True
        cboAcquisition.ListIndex = -1
        cboPlatform.ListIndex = -1
        cboSensor.ListIndex = -1
        cboGISRS.ListIndex = -1
        cboPTW.ListIndex = -1
        txtTimeRqstSent.Enabled = True
        txtTimeRqstRcvd.Enabled = True
        txtTimeInfoSent.Enabled = True
        txtTimeInfoRcvd.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datMenEvents.Enabled = False

    Else

        datMenEvents.Recordset.CancelUpdate
        cboAcquisition.Enabled = False
        cboPlatform.Enabled = False
        cboSensor.Enabled = False
        cboGISRS.Enabled = False
        cboPTW.Enabled = False
        txtTimeRqstSent.Enabled = False
        txtTimeRqstRcvd.Enabled = False
        txtTimeInfoSent.Enabled = False
        txtTimeInfoRcvd.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
        datMenEvents.Enabled = True
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess      As String
```

```
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0     'turn off error trapping

End Sub

Private Sub cmdDel_Click()

    Dim iResp       As Integer

    On Error GoTo HandleDelErrors

    If datMenEvents.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Event " & txtMensuration & "?", vbYesNo,
_
                    "Delete Event")
        If iResp = vbYes Then
            With datMenEvents.Recordset
                .Delete
                .MoveNext
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation, "Delete Event"

    End If

cmdDel_Click_Event:
    Exit Sub

HandleDelErrors:
    Dim stMess       As String

    stMess = "Cannot complete operation." & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0

End Sub

Private Sub cmdSave_Click()

    'save current record
    On Error GoTo HandleSaveErrors
    If cboAcquisition.ListIndex >= 0 And cboGISRS.ListIndex >= 0 Then
        If Val(txtCounter) < 10 Then
            txtMensuration.Text = "ME0000" & txtCounter.Text
```

```
        Else
            If Val(txtCounter) < 100 Then
                txtMensuration.Text = "ME000" & txtCounter.Text
            Else
                If Val(txtCounter) < 1000 Then
                    txtMensuration.Text = "ME00" & txtCounter.Text
                Else
                    If Val(txtCounter) < 10000 Then
                        txtMensuration.Text = "ME0" & txtCounter.Text
                    Else
                        txtMensuration.Text = "ME" & txtCounter.Text
                    End If
                End If
            End If
        End If

        datMenEvents.Recordset.Update
    Else
        MsgBox "You must select an Acquisition Event and a GISRS
Terminal before saving." _
                , vbExclamation, "Add Mensuration Event"
        datMenEvents.Recordset.CancelUpdate
    End If

    cboAcquisition.Enabled = False
    cboPlatform.Enabled = False
    cboSensor.Enabled = False
    cboGISRS.Enabled = False
    cboPTW.Enabled = False
    txtTimeRqstSent.Enabled = False
    txtTimeRqstRcvd.Enabled = False
    txtTimeInfoSent.Enabled = False
    txtTimeInfoRcvd.Enabled = False
    cmdUpdate.Enabled = True
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    cmdAdd.Caption = "&Add Event"
    mnuFile.Enabled = True
    datMenEvents.Enabled = True
    cmdAdd.SetFocus


    datMenEvents.Enabled = True

cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping
```

```
        Case 3058, 3315      'no entry in key field
            stMess = "Select Acquisition Event and GISRS Terminal
before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0      'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datMenEvents.Recordset.CancelUpdate
            Resume Next
    End Select


End Sub

Private Sub cmdUpdate_Click()

    If cmdUpdate.Caption = "&Update" And _
        datMenEvents.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboAcquisition.Enabled = True
        cboPlatform.Enabled = True
        cboSensor.Enabled = True
        cboGISRS.Enabled = True
        cboPTW.Enabled = True
        txtTimeRqstSent.Enabled = True
        txtTimeRqstRcvd.Enabled = True
        txtTimeInfoSent.Enabled = True
        txtTimeInfoRcvd.Enabled = True
        cmdAdd.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        datMenEvents.Enabled = False
        datMenEvents.Recordset.Edit

    Else
        If datMenEvents.Recordset.RecordCount > 0 Then
            datMenEvents.Recordset.Update

            cboAcquisition.Enabled = False
            cboPlatform.Enabled = False
            cboSensor.Enabled = False
            cboGISRS.Enabled = False
            cboPTW.Enabled = False
            txtTimeRqstSent.Enabled = False
            txtTimeRqstRcvd.Enabled = False
            txtTimeInfoSent.Enabled = False
            txtTimeInfoRcvd.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
```

```
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datMenEvents.Enabled = True


        End If
    End If

End Sub

Private Sub datMenEvents_Reposition()

    SetMenEventsRecordNumber


End Sub

Private Sub Form_Load()

    datMenEvents.DatabaseName = gstNewDatabase

    stSQL1 = "Select Acquisition from Acquisition"
    stSQL2 = "Select SensorType from SensorType"
    stSQL3 = "Select Platform from Platform"
    stSQL4 = "Select GISRSTerminal from GISRSTerminal"
    stSQL5 = "Select PTWTerminal from PTWTerminal"


    Set rsAcquisition = db.OpenRecordset(stSQL1)
    Set rsSensor = db.OpenRecordset(stSQL2)
    Set rsPlatform = db.OpenRecordset(stSQL3)
    Set rsGISRS = db.OpenRecordset(stSQL4)
    Set rsPTW = db.OpenRecordset(stSQL5)

    'fill cboAcquisition
    Do Until rsAcquisition.EOF
        cboAcquisition.AddItem rsAcquisition!Acquisition
        rsAcquisition.MoveNext
    Loop

    'fill cboSensor
    Do Until rsSensor.EOF
        cboSensor.AddItem rsSensor!SensorType
        rsSensor.MoveNext
    Loop

    'fill cboPlatform
    Do Until rsPlatform.EOF
        cboPlatform.AddItem rsPlatform!Platform
        rsPlatform.MoveNext
    Loop

    'fill cboGISRS
    Do Until rsGISRS.EOF
```

```
        cboGISRS.AddItem rsGISRS!GISRSTerminal
        rsGISRS.MoveNext
    Loop

    'fill cboPTW
    Do Until rsPTW.EOF
        cboPTW.AddItem rsPTW!PTWTerminal
        rsPTW.MoveNext
    Loop

    With datMenEvents
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetMenEventsRecordNumber



End Sub

Private Sub SetMenEventsRecordNumber()

    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datMenEvents.Recordset.RecordCount
    iCurrentRecord = datMenEvents.Recordset.AbsolutePosition + 1

    If datMenEvents.Recordset.EOF Then
        datMenEvents.Caption = "No more records"
    Else
        datMenEvents.Caption = "Mensuration Event Record " &
iCurrentRecord & _
                               " of " & iRecordCount
    End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub
```

```
Private Sub mnuFilePrint_Click()
    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datMenEvents.Recordset

            If bWord = True Then

                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=10


                Sel.TypeText Text:="Mensuration"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Request Sent"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Request Rcvd"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Info Sent"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Info Received"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Acquisition"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Sensor Type"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="Platform"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="GISRS"
                Sel.MoveRight unit:=12                    '12=next cell

                Sel.TypeText Text:="PTW"
                Sel.MoveRight unit:=12                    '12=next cell

                Do Until .EOF

                    Sel.TypeText Text:=!Mensuration
                    Sel.MoveRight unit:=12                '12=next
```

```
cell

                    Sel.TypeText Text:=!TimeRequestSent
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TimeRequestReceived
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TimeInfoSent
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TimeInfoReceived
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!Acquisition
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!MenSensorType
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!MenPlatform
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!GISRSTerminal
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!PTWTerminal
                    Sel.MoveRight unit:=12                  '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\MenEvents.txt" For Output As #1

                    Print #1, "Mensuration"; Chr(9); "Request Sent";
Chr(9); "Request Rcvd"; Chr(9); _
                                "Info Sent"; Chr(9); "Info Rcvd";
```

```
Chr(9); _
                                "Acquisition"; Chr(9); "Sensor Type";
Chr(9); _
                                "Platform"; Chr(9); "GISRS"; Chr(9); _
                                "PTW"; Chr(9)

                    Do Until .EOF

                        Print #1, !Mensuration; Chr(9); _
                                  !TimeRequestSent; Chr(9); _
                                  !TimeRequestReceived; Chr(9); _
                                  !TimeInfoSent; Chr(9); _
                                  !TimeInfoReceived; Chr(9); _
                                  !AcqSensorType; Chr(9); _
                                  !Acquisition; Chr(9); _
                                  !MenSensorType; Chr(9); _
                                  !MenPlatform; Chr(9); _
                                  !GISRSTerminal; Chr(9); _
                                  !PTWTerminal; Chr(9)

                        .MoveNext
                    Loop

                    Close #1

                End If
            End If

        .MoveFirst

        End With

    End If

    bContinue = False
    bWord = False
    bText = False


mnuPrintErrors:
        Select Case Err.Number
            Case 94
                Sel.TypeText Text:=""
                Resume Next
        End Select

End Sub

Private Sub txtAcquisition_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsAcquisition.MoveFirst
```

```
    If txtAcquisition <> "" Then
        Do Until iIndex = rsAcquisition.RecordCount Or bFound
            If rsAcquisition!Acquisition = txtAcquisition Then
                cboAcquisition.Text = rsAcquisition!Acquisition
                bFound = True
            Else
                rsAcquisition.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub

Private Sub txtGISRS_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsGISRS.MoveFirst
    If txtGISRS <> "" Then
        Do Until iIndex = rsGISRS.RecordCount Or bFound
            If rsGISRS!GISRSTerminal = txtGISRS Then
                cboGISRS.Text = rsGISRS!GISRSTerminal
                bFound = True
            Else
                rsGISRS.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub

Private Sub txtPlatform_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsPlatform.MoveFirst
    If txtPlatform <> "" Then
        Do Until iIndex = rsPlatform.RecordCount Or bFound
            If rsPlatform!Platform = txtPlatform Then
                cboPlatform.Text = rsPlatform!Platform
                bFound = True
            Else
                rsPlatform.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
```

```
    End If

End Sub

Private Sub txtPTW_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsPTW.MoveFirst
    If txtPTW <> "" Then
        Do Until iIndex = rsPTW.RecordCount Or bFound
            If rsPTW!PTWTerminal = txtPTW Then
                cboPTW.Text = rsPTW!PTWTerminal
                bFound = True
            Else
                rsPTW.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub

Private Sub txtSensor_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsSensor.MoveFirst
    If txtSensor <> "" Then
        Do Until iIndex = rsSensor.RecordCount Or bFound
            If rsSensor!SensorType = txtSensor Then
                cboSensor.Text = rsSensor!SensorType
                bFound = True
            Else
                rsSensor.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub


'************************************************************
'Module:         frmNomEvents.frm
'Description:    Allows user to access the nomination event
'                records for addition, deletion, and
'                modification.
```

```
'Programmer:    Kevin Colón
'*************************************************************

Option Explicit

Dim rsAcquisition   As Recordset
Dim rsMensuration   As Recordset
Dim rsGISRS         As Recordset
Dim stSQL1          As String
Dim stSQL2          As String
Dim stSQL3          As String
Private WordApp     As Word.Application
Private Doc         As Word.Document
Private Sel         As Word.Selection


Private Sub cboAcquisition_Change()

    If cboAcquisition.ListIndex >= 0 Then
        txtAcquisition = cboAcquisition.Text
    End If

End Sub

Private Sub cboGISRS_Change()

    If cboGISRS.ListIndex >= 0 Then
        txtGISRS = cboGISRS.Text
    End If

End Sub

Private Sub cboMensuration_Change()

    If cboMensuration.ListIndex >= 0 Then
        txtMensuration = cboMensuration.Text
    End If

End Sub

Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add Event" Then

        datNomination.Recordset.AddNew
        cboAcquisition.Enabled = True
        cboMensuration.Enabled = True
        cboGISRS.Enabled = True
        cboAcquisition.ListIndex = -1
        cboMensuration.ListIndex = -1
        cboGISRS.ListIndex = -1
        txtTimeSent.Enabled = True
        txtTimeRcvd.Enabled = True
```

```
        txtAssess.Enabled = True
        txtTLE.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datNomination.Enabled = False

    Else

        datNomination.Recordset.CancelUpdate
        cboAcquisition.Enabled = False
        cboMensuration.Enabled = False
        cboGISRS.Enabled = False
        txtTimeSent.Enabled = False
        txtTimeRcvd.Enabled = False
        txtAssess.Enabled = False
        txtTLE.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
        datNomination.Enabled = True
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess      As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0     'turn off error trapping

End Sub

Private Sub cmdDel_Click()

    Dim iResp       As Integer

    On Error GoTo HandleDelErrors

    If datNomination.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Event " & txtNomination & "?", vbYesNo,
_
                    "Delete Event")
        If iResp = vbYes Then
            With datNomination.Recordset
                .Delete
                .MoveNext
```

```
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation, "Delete Event"

    End If

cmdDel_Click:
    Exit Sub

HandleDelErrors:
    Dim stMess       As String

    stMess = "Cannot complete operation." & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0


End Sub

Private Sub cmdSave_Click()

    'save current record
    On Error GoTo HandleSaveErrors
    If cboAcquisition.ListIndex >= 0 And cboGISRS.ListIndex >= 0 Then
        If Val(txtCounter) < 10 Then
            txtNomination.Text = "NE0000" & txtCounter.Text
        Else
            If Val(txtCounter) < 100 Then
                txtNomination.Text = "NE000" & txtCounter.Text
            Else
                If Val(txtCounter) < 1000 Then
                    txtNomination.Text = "NE00" & txtCounter.Text
                Else
                    If Val(txtCounter) < 10000 Then
                        txtNomination.Text = "NE0" & txtCounter.Text
                    Else
                        txtNomination.Text = "NE" & txtCounter.Text
                    End If
                End If
            End If
        End If

        datNomination.Recordset.Update
    Else
        MsgBox "You must select an Acquisition Event and a GISRS
Terminal before saving." _
```

```
                , vbExclamation, "Add Nomination Event"
            datNomination.Recordset.CancelUpdate
        End If

        cboAcquisition.Enabled = False
        cboMensuration.Enabled = False
        cboGISRS.Enabled = False
        txtTLE.Enabled = False
        txtTimeSent.Enabled = False
        txtTimeRcvd.Enabled = False
        txtAssess.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add Event"
        mnuFile.Enabled = True
        datNomination.Enabled = True
        cmdAdd.SetFocus


        datNomination.Enabled = True

cmdSave_Click_Exit:
        Exit Sub

HandleSaveErrors:
        Dim stMess  As String
        Select Case Err.Number
            Case 3022           'duplicate key field
                stMess = "Record already exists -- could not save>'"
                MsgBox stMess, vbExclamation, "Database Error"
                On Error GoTo 0     'turn off error trapping

            Case 3058, 3315     'no entry in key field
                stMess = "Select Acquisition Event and GISRS Terminal
before saving."
                MsgBox stMess, vbExclamation, "Database Error"
                On Error GoTo 0     'turn off error trapping

            Case Else
                stMess = "Record could not be saved." & vbCrLf _
                        & Err.Description
                MsgBox stMess, vbExclamation, "Database Error"
                datNomination.Recordset.CancelUpdate
                Resume Next
        End Select

End Sub

Private Sub cmdUpdate_Click()

        If cmdUpdate.Caption = "&Update" And _
            datNomination.Recordset.RecordCount > 0 Then

            cmdUpdate.Caption = "Su&bmit"
```

```
        cboAcquisition.Enabled = True
        cboMensuration.Enabled = True
        cboGISRS.Enabled = True
        txtTimeSent.Enabled = True
        txtTimeRcvd.Enabled = True
        txtTLE.Enabled = True
        txtAssess.Enabled = True
        cmdAdd.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        datNomination.Enabled = False
        datNomination.Recordset.Edit

    Else
        If datNomination.Recordset.RecordCount > 0 Then
            datNomination.Recordset.Update

            cboAcquisition.Enabled = False
            cboMensuration.Enabled = False
            cboGISRS.Enabled = False
            txtTLE.Enabled = False
            txtTimeSent.Enabled = False
            txtTimeRcvd.Enabled = False
            txtAssess.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datNomination.Enabled = True


        End If
    End If

End Sub

Private Sub datNomination_Reposition()

    SetNominationRecordNumber

End Sub

Private Sub Form_Load()

    datNomination.DatabaseName = gstNewDatabase

    stSQL1 = "Select Acquisition from Acquisition"
    stSQL2 = "Select Mensuration from Mensuration"
    stSQL3 = "Select GISRSTerminal from GISRSTerminal"


    Set rsAcquisition = db.OpenRecordset(stSQL1)
    Set rsMensuration = db.OpenRecordset(stSQL2)
```

```
    Set rsGISRS = db.OpenRecordset(stSQL3)

    'fill cboAcquisition
    Do Until rsAcquisition.EOF
        cboAcquisition.AddItem rsAcquisition!Acquisition
        rsAcquisition.MoveNext
    Loop

    'fill cboMensuration
    Do Until rsMensuration.EOF
        cboMensuration.AddItem rsMensuration!Mensuration
        rsMensuration.MoveNext
    Loop

    'fill cboGISRS
    Do Until rsGISRS.EOF
        cboGISRS.AddItem rsGISRS!GISRSTerminal
        rsGISRS.MoveNext
    Loop

    With datNomination
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetNominationRecordNumber

End Sub

Private Sub SetNominationRecordNumber()

    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datNomination.Recordset.RecordCount
    iCurrentRecord = datNomination.Recordset.AbsolutePosition + 1

    If datNomination.Recordset.EOF Then
        datNomination.Caption = "No more records"
    Else
        datNomination.Caption = "Nomination Event Record " &
iCurrentRecord & _
                                " of " & iRecordCount
    End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me
```

```
End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFilePrint_Click()
    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datNomination.Recordset

            If bWord = True Then

                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=8

                Sel.TypeText Text:="Nomination"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Nomination Sent"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Nomination Rcvd"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Acquisition"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Mensuration"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="GISRS"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Assessment"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Target Location Error"
                Sel.MoveRight unit:=12                  '12=next cell

                Do Until .EOF
```

```
                    Sel.TypeText Text:=!Nomination
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!NomTimeSent
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!NomTimeRcvd
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!Acquisition
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!Mensuration
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!GISRSTerminal
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!Assessment
                    Sel.MoveRight unit:=12                    '12=next
cell

                    Sel.TypeText Text:=!TargetLocationError
                    Sel.MoveRight unit:=12                    '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\NomEvents.txt" For Output As #1

                    Print #1, "Nomination"; Chr(9); "Nomination Sent";
Chr(9); "Nomination Rcvd"; Chr(9); _
                                "Acquisition"; Chr(9); "Mensuration";
Chr(9); _
                                "GISRTerminal"; Chr(9); "Assessment";
Chr(9); _
                                "TargetLocationError"; Chr(9)

                    Do Until .EOF
```

```
                        Print #1, !Nomination; Chr(9); _
                                  !NomTimeSent; Chr(9); _
                                  !NomTimeRcvd; Chr(9); _
                                  !Acquisition; Chr(9); _
                                  !Mensuration; Chr(9); _
                                  !GISRSTerminal; Chr(9); _
                                  !Assessment; Chr(9); _
                                  !TargetLocationError; Chr(9)

                            .MoveNext
                        Loop

                        Close #1

                    End If
                End If

            .MoveFirst

            End With

        End If

        bContinue = False
        bWord = False
        bText = False


mnuPrintErrors:
            Select Case Err.Number
                Case 94
                    Sel.TypeText Text:=""
                    Resume Next
            End Select

End Sub

Private Sub txtAcquisition_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsAcquisition.MoveFirst
    If txtAcquisition <> "" Then
        Do Until iIndex = rsAcquisition.RecordCount Or bFound
            If rsAcquisition!Acquisition = txtAcquisition Then
                cboAcquisition.Text = rsAcquisition!Acquisition
                bFound = True
            Else
                rsAcquisition.MoveNext
                iIndex = iIndex + 1
            End If
```

```
        Loop
    End If

End Sub

Private Sub txtGISRS_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsGISRS.MoveFirst
    If txtGISRS <> "" Then
        Do Until iIndex = rsGISRS.RecordCount Or bFound
            If rsGISRS!GISRSTerminal = txtGISRS Then
                cboGISRS.Text = rsGISRS!GISRSTerminal
                bFound = True
            Else
                rsGISRS.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub

Private Sub txtMensuration_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsMensuration.MoveFirst
    If txtMensuration <> "" Then
        Do Until iIndex = rsMensuration.RecordCount Or bFound
            If rsMensuration!Mensuration = txtMensuration Then
                cboMensuration.Text = rsMensuration!Mensuration
                bFound = True
            Else
                rsMensuration.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub


'*************************************************************
'Module:        frmObjectives.frm
'Description:   Allows user to access the objectives
'               records for addition, deletion, and
```

```
'                modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datObjectives.Recordset.AddNew
        txtObjective.Enabled = True
        txtObjective.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datObjectives.Enabled = False

    Else
        datObjectives.Recordset.CancelUpdate
        txtObjective.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datObjectives.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping


End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datObjectives.Recordset.RecordCount > 0 Then
```

```
        iResp = MsgBox("Delete Objective " & txtObjective.Text & "?",
vbYesNo, "Delete Objective")
        If iResp = vbYes Then
            With datObjectives.Recordset
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Objective"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtObjective <> "" And txtDescription <> "" Then
        txtObjective = UCase(txtObjective)
        iResp = MsgBox("Do you want to add " & txtObjective & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Objective")
        If iResp = vbYes Then
            datObjectives.Recordset.Update
        End If

    Else
        MsgBox "You must enter an Objective and a description before
saving.", vbExclamation _
                , "Add Objective"
        datObjectives.Recordset.CancelUpdate
    End If
```

```
    txtObjective.Enabled = False
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datObjectives.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a Objective name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datObjectives.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datObjectives.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtObjective.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtObjective.SetFocus
        cmdAdd.Enabled = False
        datObjectives.Enabled = False
        datObjectives.Recordset.Edit
    Else
        If datObjectives.Recordset.RecordCount > 0 Then
            datObjectives.Recordset.Update

            txtObjective.Enabled = False
```

```
            txtDescription.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datObjectives.Enabled = True
        End If
    End If

End Sub

Private Sub datObjectives_Reposition()
    SetObjectiveRecordNumber

End Sub

Private Sub Form_Load()
    datObjectives.DatabaseName = gstNewDatabase

    With datObjectives
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetObjectiveRecordNumber
'

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datObjectives.Recordset.FindFirst "[Description] = '" & _
                InputBox("Enter the Objective", "Objective Search") &
"'"

    If datObjectives.Recordset.NoMatch Then
        MsgBox "Objective was not found.", vbOKOnly, "Objective Search"
```

```
        datObjectives.Recordset.MoveFirst          'go to first record
    End If

End Sub

Private Sub SetObjectiveRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datObjectives.Recordset.RecordCount
    iCurrentRecord = datObjectives.Recordset.AbsolutePosition + 1
    If datObjectives.Recordset.EOF Then
        datObjectives.Caption = "No more records"
    Else
        datObjectives.Caption = "Objective " & iCurrentRecord & _
                                " of " & iRecordCount
    End If

End Sub



'************************************************************
'Module:         frmPlatforms.frm
'Description:    Allows user to access the platform
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Private Sub cboPlatType_Click()

    If cboPlatType.ListIndex >= 0 Then
        txtPlatTypeId = cboPlatType.Text

    End If

End Sub

Private Sub cmdAddPlat_Click()

    On Error GoTo HandleAddPlatErrors

    If cmdAddPlat.Caption = "&Add Platform" Then

        datPlatforms.Recordset.AddNew
        cboPlatType.Enabled = True
        cboPlatType.ListIndex = 0
        txtPlatform.Enabled = True
        txtPlatName.Enabled = True
        txtCommander.Enabled = True
        txtSpecialty.Enabled = True
        txtLocation.Enabled = True
```

```
        txtLogger.Enabled = True
        txtPlatTypeId.Enabled = True
        cmdSavePlat.Enabled = True
        cmdDelPlat.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        cboPlatType.SetFocus
        cmdAddPlat.Caption = "&Cancel"
        datPlatforms.Enabled = False
    Else
        datPlatforms.Recordset.CancelUpdate

        cboPlatType.Enabled = False
        txtPlatform.Enabled = False
        txtPlatName.Enabled = False
        txtCommander.Enabled = False
        txtSpecialty.Enabled = False
        txtLocation.Enabled = False
        txtLogger.Enabled = False
        txtPlatTypeId.Enabled = False
        cmdSavePlat.Enabled = False
        cmdDelPlat.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAddPlat.Caption = "&Add Platform"
        cmdAddPlat.SetFocus

        datPlatforms.Enabled = True


    End If

cmdAddPlat_Click_Exit:
    Exit Sub

HandleAddPlatErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping


End Sub

Private Sub FillPlatTypeCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
    cboPlatType.Clear

    With datPlatType
        .Refresh        'open database
        iCount = .Recordset.RecordCount
```

```
        'fill the list
        Do Until .Recordset.EOF
            If .Recordset!PlatformType <> "" Then
                cboPlatType.AddItem .Recordset!PlatformType

            End If
            .Recordset.MoveNext
        Loop
    End With

End Sub

Private Sub cmdDelPlat_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelPlatErrors

    If datPlatforms.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Platform " & txtPlatform & "?", vbYesNo,
_
                "Delete Platform")
        If iResp = vbYes Then
            With datPlatforms.Recordset
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Event"

    End If

cmdDelPlat_Click_Exit:
    Exit Sub

HandleDelPlatErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSavePlat_Click()
```

```
    'save the current record

    On Error GoTo HandleSavePlatformErrors
    If cboPlatType.ListIndex >= 0 Then
        If txtPlatform <> "" And txtPlatName <> "" Then
            datPlatforms.Recordset.Update
        Else
            MsgBox "You must enter a Platform name and id before
saving." _
                    , vbExclamation, "Add Platform"
            datPlatforms.Recordset.CancelUpdate
        End If
    Else
            MsgBox "You must select a Platform Type before saving." _
                    , vbExclamation, "Add Platform"
            datPlatforms.Recordset.CancelUpdate
    End If

    cboPlatType.Enabled = False
    txtPlatform.Enabled = False
    txtPlatName.Enabled = False
    txtCommander.Enabled = False
    txtSpecialty.Enabled = False
    txtLocation.Enabled = False
    txtLogger.Enabled = False
    txtPlatTypeId.Enabled = False
    cmdSavePlat.Enabled = False
    cmdDelPlat.Enabled = True
    cmdUpdate.Enabled = True
    mnuFile.Enabled = True
    cmdAddPlat.Caption = "&Add Platform"
    cmdAddPlat.SetFocus

    datPlatforms.Enabled = True

cmdSavePlat_Click_Exit:
    Exit Sub

HandleSavePlatformErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Select a platform type before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
```

```
            datPlatforms.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datPlatforms.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboPlatType.Enabled = True
        txtPlatform.Enabled = True
        txtPlatform.SetFocus
        txtPlatName.Enabled = True
        txtCommander.Enabled = True
        txtSpecialty.Enabled = True
        txtLocation.Enabled = True
        txtLogger.Enabled = True
        txtPlatTypeId.Enabled = True
        cmdDelPlat.Enabled = False
        cmdAddPlat.Enabled = False
        mnuFile.Enabled = False
        datPlatforms.Enabled = False
        datPlatforms.Recordset.Edit

    Else
        If datPlatforms.Recordset.RecordCount > 0 Then
            datPlatforms.Recordset.Update

            cboPlatType.Enabled = False
            txtPlatform.Enabled = False
            txtPlatName.Enabled = False
            txtCommander.Enabled = False
            txtSpecialty.Enabled = False
            txtLocation.Enabled = False
            txtLogger.Enabled = False
            txtPlatTypeId.Enabled = False
            cmdDelPlat.Enabled = True
            mnuFile.Enabled = True
            cmdAddPlat.Enabled = True
            cmdAddPlat.SetFocus
            cmdUpdate.Caption = "&Update"
            datPlatforms.Enabled = True
        End If
    End If

End Sub

Private Sub datPlatforms_Reposition()

    SetPlatformRecordNumber

End Sub
```

```
Private Sub Form_Load()
    datPlatforms.DatabaseName = gstNewDatabase
    datPlatType.DatabaseName = gstNewDatabase

    FillPlatTypeCombo

    With datPlatforms
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    datPlatType.Refresh
    datPlatType.Recordset.MoveFirst

    SetPlatformRecordNumber

End Sub

Private Sub SetPlatformRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datPlatforms.Recordset.RecordCount
    iCurrentRecord = datPlatforms.Recordset.AbsolutePosition + 1
    If datPlatforms.Recordset.EOF Then
        datPlatforms.Caption = "No more records"
    Else
        datPlatforms.Caption = "Platform Record " & iCurrentRecord & _
                               " of " & iRecordCount
    End If

End Sub


Private Sub mnuFileBack_Click()

    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datPlatforms.Recordset.FindFirst "[PlatformId] = '" & _
                InputBox("Enter the Platform Id", "Platform Id Search")
& "'"

    If datPlatforms.Recordset.NoMatch Then
        MsgBox "Platform Id was not found.", vbOKOnly, "Platform Id
Search"
        datPlatforms.Recordset.MoveFirst         'go to first record
```

```
    End If

End Sub

Private Sub txtPlatTypeId_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
    Dim bFound  As Boolean

    datPlatType.Recordset.MoveFirst
    If txtPlatTypeId <> "" Then
        Do Until iIndex = datPlatType.Recordset.RecordCount Or bFound
            If datPlatType.Recordset!PlatformType = txtPlatTypeId Then
                cboPlatType.Text = datPlatType.Recordset!PlatformType
                bFound = True
            Else
                datPlatType.Recordset.MoveNext
                iIndex = iIndex = 1

            End If
        Loop
    Else
        cboPlatType.ListIndex = -1
    End If

End Sub



'*************************************************************
'Module:         frmPlatformsTypes.frm
'Description:    Allows user to access the platform types
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datPlatTypes.Recordset.AddNew
        txtPlatformType.Enabled = True
        txtPlatformType.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datPlatTypes.Enabled = False
```

```
    Else
        datPlatTypes.Recordset.CancelUpdate
        txtPlatformType.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datPlatTypes.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datPlatTypes.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Platform " & txtPlatformType.Text & "?",
vbYesNo, "Delete Platform")
        If iResp = vbYes Then
            With datPlatTypes.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Platform"

    End If

cmdDel_Click_Exit:
```

```
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtPlatformType.Text <> "" Then
        txtPlatformType.Text = UCase(txtPlatformType.Text)
        iResp = MsgBox("Do you want to add " & txtPlatformType.Text & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Platform")
        If iResp = vbYes Then
            datPlatTypes.Recordset.Update
        End If

    Else
        MsgBox "You must enter a Platform type before saving.",
vbExclamation _
                , "Add Platform"
        datPlatTypes.Recordset.CancelUpdate
    End If

    txtPlatformType.Enabled = False
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datPlatTypes.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping
```

```
        Case 3058, 3315      'no entry in key field
            stMess = "Enter a Platform type before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0      'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datPlatTypes.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datPlatTypes.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtPlatformType.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtPlatformType.SetFocus
        cmdAdd.Enabled = False
        datPlatTypes.Enabled = False
        datPlatTypes.Recordset.Edit
    Else
        If datPlatTypes.Recordset.RecordCount > 0 Then
            datPlatTypes.Recordset.Update

            txtPlatformType.Enabled = False
            txtDescription.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datPlatTypes.Enabled = True
        End If
    End If

End Sub

Private Sub datPlattypes_Reposition()

    SetPlatformRecordNumber

End Sub

Private Sub Form_Load()

    datPlatTypes.DatabaseName = gstNewDatabase
```

```
    With datPlatTypes
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetPlatformRecordNumber


End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datPlatTypes.Recordset.FindFirst "[PlatformType] = '" & _
                InputBox("Enter the Platform Type", "Platform Type
Search") & "'"

    If datPlatTypes.Recordset.NoMatch Then
        MsgBox "Platform Type was not found.", vbOKOnly, "Platform Type
Search"
        datPlatTypes.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetPlatformRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datPlatTypes.Recordset.RecordCount
    iCurrentRecord = datPlatTypes.Recordset.AbsolutePosition + 1
    If datPlatTypes.Recordset.EOF Then
        datPlatTypes.Caption = "No more records"
    Else
        datPlatTypes.Caption = "Platform " & iCurrentRecord & _
                               " of " & iRecordCount
    End If

End Sub
```

```
'*************************************************************
'Module:         frmPrint.frm
'Description:    Allows user to choose to export report to Word
'                or to a text file
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

Private Sub cmdCancel_Click()

    Unload Me

End Sub

Private Sub cmdOK_Click()

    bContinue = True
    If Option1.Value = True Then
        bWord = True
    Else
        If Option2.Value = True Then
            bText = True
        End If
    End If
    Unload Me

End Sub



'*************************************************************
'Module:         frmPTW.frm
'Description:    Allows user to access the PTW terminal
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit
Dim rsPlatform      As Recordset
Dim stSQL           As String

Private Sub cboPlatform_Click()

    If cboPlatform.ListIndex >= 0 Then

        txtPlatform = cboPlatform.Text

    End If

End Sub
```

```
Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datPTW.Recordset.AddNew
        txtTerminal.Enabled = True
        txtTerminal.SetFocus
        txtFunction.Enabled = True
        cboPlatform.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datPTW.Enabled = False

    Else
        datPTW.Recordset.CancelUpdate
        txtTerminal.Enabled = False
        txtFunction.Enabled = False
        cboPlatform.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datPTW.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
           & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datPTW.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Terminal " & txtTerminal.Text & "?",
vbYesNo, "Delete Terminal")
        If iResp = vbYes Then
            With datPTW.Recordset
```

```
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Terminal"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtTerminal.Text <> "" Then
        txtTerminal.Text = UCase(txtTerminal.Text)
        iResp = MsgBox("Do you want to add " & txtTerminal.Text & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Terminal")
        If iResp = vbYes Then
            datPTW.Recordset.Update
        End If

    Else
        MsgBox "You must enter a Terminal before saving.",
vbExclamation _
                , "Add Terminal"
        datPTW.Recordset.CancelUpdate
    End If

    txtTerminal.Enabled = False
    txtFunction.Enabled = False
    cboPlatform.Enabled = False
    cmdSave.Enabled = False
```

```
    cmdDel.Enabled = True
    datPTW.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a location before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datPTW.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datPTW.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtTerminal.Enabled = True
        txtFunction.Enabled = True
        cboPlatform.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtTerminal.SetFocus
        cmdAdd.Enabled = False
        datPTW.Enabled = False
        datPTW.Recordset.Edit
    Else
        If datPTW.Recordset.RecordCount > 0 Then
            txtTerminal = UCase(txtTerminal)
            datPTW.Recordset.Update

            txtTerminal.Enabled = False
            txtFunction.Enabled = False
```

```
            cboPlatform.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datPTW.Enabled = True
        End If
    End If

End Sub

Private Sub datPTW_Reposition()

    SetTerminalRecordNumber

End Sub

Private Sub FillPlatformCombo()

    Dim iCount As Integer
    'fill the PlatType combo box
    cboPlatform.Clear

    With rsPlatform

        iCount = .RecordCount

        'fill the list
        Do Until .EOF
            If !Platform <> "" Then
                cboPlatform.AddItem !Platform

            End If
            .MoveNext
        Loop
    End With


End Sub

Private Sub Form_Load()

    datPTW.DatabaseName = gstNewDatabase

    stSQL = "Select Platform from Platform"

    Set rsPlatform = db.OpenRecordset(stSQL)

    FillPlatformCombo

    With datPTW
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
```

```
            .Recordset.MoveFirst
        End If
    End With

    SetTerminalRecordNumber

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datPTW.Recordset.FindFirst "[PTWTerminal] = '" & _
                InputBox("Enter the PTW Terminal", "PTW Terminal
Search") & "'"

    If datPTW.Recordset.NoMatch Then
        MsgBox "PTW Terminal was not found.", vbOKOnly, "PTW Terminal
Search"
        datPTW.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetTerminalRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datPTW.Recordset.RecordCount
    iCurrentRecord = datPTW.Recordset.AbsolutePosition + 1
    If datPTW.Recordset.EOF Then
        datPTW.Caption = "No more records"
    Else
        datPTW.Caption = "Terminal " & iCurrentRecord & _
                            " of " & iRecordCount
    End If

End Sub

Private Sub txtPlatform_Change()

    'selects correct combo box listing
    Dim iIndex  As Integer
```

```
    Dim bFound  As Boolean

    rsPlatform.MoveFirst
    If txtPlatform <> "" Then
        Do Until iIndex = rsPlatform.RecordCount Or bFound
            If rsPlatform!Platform = txtPlatform Then
                cboPlatform.Text = rsPlatform!Platform
                bFound = True
            Else
                rsPlatform.MoveNext
                iIndex = iIndex + 1
            End If
        Loop
    Else
        cboPlatform.ListIndex = -1
    End If

End Sub


'************************************************************
'Module:        frmQueries.frm
'Description:   Contains predefined queries that filter
'               acquisitions and targets based on a parameter
'               selected by the user.
'Programmer:    Kevin Colón
'************************************************************

Option Explicit
Dim Index   As Integer

Private Sub cmdCancel_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub cmdSubmit_Click()

    frmQueryOutput.Show
    Me.Enabled = False

End Sub

Private Sub Form_Load()

    datPlatforms.DatabaseName = gstNewDatabase
    datSensTypes.DatabaseName = gstNewDatabase
    datWeaponTypes.DatabaseName = gstNewDatabase
    datThreatTypes.DatabaseName = gstNewDatabase

    'fill the Platform combo box
    FillPlatforms
```

```
    'fill the Sensor combo box
    FillSensTypes

    'fill the Threat combo box
    FillThreatTypes

    'fill the Weapon combo box
    FillWeaponTypes

End Sub

Private Sub FillPlatforms()

    cboPlatforms1.Clear
    cboPlatforms2.Clear

    With datPlatforms
        .Refresh        'open database
        'fill the list
        Do Until .Recordset.EOF     'until no more records in recordset
            If .Recordset!Platform <> "" Then
                cboPlatforms1.AddItem .Recordset!Platform
                cboPlatforms2.AddItem .Recordset!Platform

            End If
            .Recordset.MoveNext
        Loop
    End With

End Sub

Private Sub FillSensTypes()

    cboSensTypes1.Clear
    cboSensTypes2.Clear

    With datSensTypes
        .Refresh        'open database
        'fill the list
        Do Until .Recordset.EOF
            If .Recordset!SensorType <> "" Then
                cboSensTypes1.AddItem .Recordset!SensorType
                cboSensTypes2.AddItem .Recordset!SensorType

            End If
            .Recordset.MoveNext
        Loop
    End With

End Sub

Private Sub FillThreatTypes()

    cboThreatTypes.Clear
```

```
    With datThreatTypes
        .Refresh        'open database
        'fill the list
        Do Until .Recordset.EOF
            If .Recordset!ThreatType <> "" Then
                cboThreatTypes.AddItem .Recordset!ThreatType
            End If
            .Recordset.MoveNext
        Loop
    End With

End Sub

Private Sub FillWeaponTypes()

    cboWeaponTypes.Clear

    With datWeaponTypes
        .Refresh        'open database
        'fill the list
        Do Until .Recordset.EOF
            If .Recordset!WeaponType <> "" Then
                cboWeaponTypes.AddItem .Recordset!WeaponType
            End If
            .Recordset.MoveNext
        Loop
    End With

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub Label1_Click()

    optQuery(0).Value = True

End Sub

Private Sub Label10_Click()

    optQuery(9).Value = True

End Sub

Private Sub Label11_Click()

    optQuery(2).Value = True

End Sub

Private Sub Label2_Click()
```

```
    optQuery(1).Value = True
End Sub
Private Sub Label3_Click()
    optQuery(2).Value = True
End Sub
Private Sub Label4_Click()
    optQuery(3).Value = True
End Sub
Private Sub Label5_Click()
    optQuery(4).Value = True
End Sub
Private Sub Label6_Click()
    optQuery(5).Value = True
End Sub
Private Sub Label7_Click()
    optQuery(6).Value = True
End Sub
Private Sub Label8_Click()
    optQuery(7).Value = True
End Sub
Private Sub Label9_Click()
    optQuery(8).Value = True
End Sub
Private Sub mnuFileBack_Click()
    frmMain.Enabled = True
    Unload Me
End Sub
Private Sub optQuery_Click(Index As Integer)
```

```
Select Case Index
    Case 0
        cboPlatforms1.Enabled = True
        cboSensTypes1.Enabled = False
        cboPlatforms2.Enabled = False
        cboSensTypes2.Enabled = False
        cboWeaponTypes.Enabled = False
        cboThreatTypes.Enabled = False

    Case 1
        cboPlatforms1.Enabled = False
        cboSensTypes1.Enabled = True
        cboPlatforms2.Enabled = False
        cboSensTypes2.Enabled = False
        cboWeaponTypes.Enabled = False
        cboThreatTypes.Enabled = False

    Case 2
        cboPlatforms1.Enabled = False
        cboSensTypes1.Enabled = False
        cboPlatforms2.Enabled = True
        cboSensTypes2.Enabled = True
        cboWeaponTypes.Enabled = False
        cboThreatTypes.Enabled = False

    Case 3
        cboPlatforms1.Enabled = False
        cboSensTypes1.Enabled = False
        cboPlatforms2.Enabled = False
        cboSensTypes2.Enabled = False
        cboWeaponTypes.Enabled = True
        cboThreatTypes.Enabled = False

    Case 4
        cboPlatforms1.Enabled = False
        cboSensTypes1.Enabled = False
        cboPlatforms2.Enabled = False
        cboSensTypes2.Enabled = False
        cboWeaponTypes.Enabled = False
        cboThreatTypes.Enabled = True

    Case 5
        cboPlatforms1.Enabled = False
        cboSensTypes1.Enabled = False
        cboPlatforms2.Enabled = False
        cboSensTypes2.Enabled = False
        cboWeaponTypes.Enabled = False
        cboThreatTypes.Enabled = False

    Case 6
        cboPlatforms1.Enabled = False
        cboSensTypes1.Enabled = False
        cboPlatforms2.Enabled = False
        cboSensTypes2.Enabled = False
        cboWeaponTypes.Enabled = False
```

```
            cboThreatTypes.Enabled = False

        Case 7
            cboPlatforms1.Enabled = False
            cboSensTypes1.Enabled = False
            cboPlatforms2.Enabled = False
            cboSensTypes2.Enabled = False
            cboWeaponTypes.Enabled = False
            cboThreatTypes.Enabled = False

        Case 8
            cboPlatforms1.Enabled = False
            cboSensTypes1.Enabled = False
            cboPlatforms2.Enabled = False
            cboSensTypes2.Enabled = False
            cboWeaponTypes.Enabled = False
            cboThreatTypes.Enabled = False

        Case 9
            cboPlatforms1.Enabled = False
            cboSensTypes1.Enabled = False
            cboPlatforms2.Enabled = False
            cboSensTypes2.Enabled = False
            cboWeaponTypes.Enabled = False
            cboThreatTypes.Enabled = False

    End Select

End Sub


'*************************************************************
'Module:         frmQueryOutput.frm
'Description:    Displays the results from the query executed
'                from the Queries form (frmQueries)
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit


Private Sub Form_Load()
    Dim stSQL           As String
    Dim stPrev          As String
    Dim RS              As Recordset
    Dim iIndex          As Integer
    Dim iRecord         As Integer
    Dim rsNominations   As Recordset
    Dim rsTargets       As Recordset
    Dim stNoms          As String
    Dim stTargets       As String
    Dim bFound          As Boolean

    datQuery.DatabaseName = gstNewDatabase
```

```
    'Query by platform used for detection
    If frmQueries.optQuery(0).Value = True Then
        stSQL = "Select * from Acquisition " & _
                "Where Acquisition.AcqPlatform = '" & _
                frmQueries.cboPlatforms1.Text & "'"

        Set RS = db.OpenRecordset(stSQL)
        FlexOutput.FormatString = "Record|Acquistion Event|Track
Id|Platform|Time of Acquisition     "

        Do Until RS.EOF
            iRecord = iRecord + 1

            FlexOutput.AddItem iRecord & vbTab & RS!Acquisition & vbTab
& _
                RS!TrackId & vbTab & RS!AcqPlatform & vbTab &
RS!AcqTime
            RS.MoveNext

        Loop
    End If


    'Query by sensor used for detection
    If frmQueries.optQuery(1).Value = True Then
        stSQL = "Select * from Acquisition, SensorType " & _
                "Where Acquisition.AcqSensorType = '" & _
                frmQueries.cboSensTypes1.Text & _
                "' And SensorType.SensorType = '" & _
                frmQueries.cboSensTypes1.Text & "'"

        Set RS = db.OpenRecordset(stSQL)

        FlexOutput.FormatString = "Record|Acquistion Event|Track
Id|Sensor Type|Time of Acquisition     "
        Do Until RS.EOF
            iRecord = iRecord + 1
            FlexOutput.AddItem iRecord & vbTab & RS!Acquisition & vbTab
& RS!TrackId & vbTab _
                & frmQueries.cboSensTypes1.Text & vbTab & RS!AcqTime
            RS.MoveNext
        Loop
    End If


    'Query by sensor and platform detection
    If frmQueries.optQuery(2).Value = True Then
        stSQL = "Select * from Acquisition, SensorType " & _
                "Where Acquisition.AcqSensorType = '" & _
                frmQueries.cboSensTypes2.Text & _
```

```
                "' And Acquisition.AcqPlatform = '" & _
                frmQueries.cboPlatforms2.Text & _
                "' And SensorType.SensorType = '" & _
                frmQueries.cboSensTypes2.Text & "'"


        Set RS = db.OpenRecordset(stSQL)

        FlexOutput.FormatString = "Record|Acquistion Event|Track
Id|Platform |Sensor Type|Time of Acquisition     "
        Do Until RS.EOF
            iRecord = iRecord + 1
            FlexOutput.AddItem iRecord & vbTab & RS!Acquisition & vbTab
& RS!TrackId & vbTab _
                & RS!AcqPlatform & vbTab &
frmQueries.cboSensTypes2.Text & vbTab & RS!AcqTime
            RS.MoveNext
        Loop
    End If


    'Query by weapon used
    If frmQueries.optQuery(3).Value = True Then
        stSQL = "Select * from Target, WeaponType " & _
                "Where Target.WeaponType = '" & _
                frmQueries.cboWeaponTypes.Text & _
                "' And WeaponType.WeaponType = '" & _
                frmQueries.cboWeaponTypes.Text & "'"

        Set RS = db.OpenRecordset(stSQL)
        FlexOutput.FormatString = "Record|Target Id|Time Target
Designated|Sensor Type|NLT Time                  "

        Do Until RS.EOF
            iRecord = iRecord + 1
            FlexOutput.AddItem iRecord & vbTab & RS!TargetId & vbTab &
RS!TimeofDesignation _
                & vbTab & frmQueries.cboWeaponTypes.Text & vbTab &
RS!TargetNLTTime
            RS.MoveNext
        Loop
    End If


    'Query by threat types detected
    If frmQueries.optQuery(4).Value = True Then
        stSQL = "Select * from Acquisition, ThreatType " & _
                "Where Acquisition.ThreatType = '" & _
                frmQueries.cboThreatTypes.Text & _
                "' And ThreatType.ThreatType = '" & _
                frmQueries.cboThreatTypes.Text & "'"
        Set RS = db.OpenRecordset(stSQL)

        FlexOutput.FormatString = "Record|Acquistion Event|Track
Id|Threat Type       |Time of Acquisition     "
```

```
        Do Until RS.EOF
            iRecord = iRecord + 1
            FlexOutput.AddItem iRecord & vbTab & RS!Acquisition & vbTab
& RS!TrackId & vbTab _
                & frmQueries.cboThreatTypes.Text & vbTab & RS!AcqTime
            RS.MoveNext
        Loop
    End If


    'Average time acquisition to mensuration
    If frmQueries.optQuery(5).Value = True Then


    End If


    'Average time fire command to fire event
    If frmQueries.optQuery(6).Value = True Then


    End If


    'Nominations accepted as targets
    If frmQueries.optQuery(7).Value = True Then
        stSQL = "SELECT Nomination.Nomination, Target.TargetId,
Target.Description, " & _
                "Nomination.Acquisition, Nomination.Mensuration " & _
                "From Nomination, Target " & _
                "WHERE Nomination.Nomination = Target.Nomination"

        Set RS = db.OpenRecordset(stSQL)

        FlexOutput.FormatString = "Record|Nomination |Target
Id|Description       |Acquisition    |Mensuration    "
        FlexOutput.Clear
        iRecord = 1
        Do Until RS.EOF
            FlexOutput.AddItem iRecord & vbTab & RS!Nomination & vbTab
& _
                RS!TargetId & vbTab & RS!Description & vbTab &
RS!Acquisition & _
                vbTab & RS!Mensuration
            stPrev = RS!Nomination
            RS.MoveNext
            If Not RS.EOF Then
                If stPrev <> RS!Nomination Then
                    iRecord = iRecord + 1
                End If
            End If
        Loop
```

```
    End If


    'Nominations declined as targets
    If frmQueries.optQuery(8).Value = True Then
        stNoms = "Select * " & _
                 "From Nomination "
        stTargets = "Select * " & _
                    "From Target "
        Set rsNominations = db.OpenRecordset(stNoms)
        Set rsTargets = db.OpenRecordset(stTargets)
        datQuery.DatabaseName = gstNewDatabase

        rsNominations.MoveFirst

        iRecord = 1
        FlexOutput.FormatString = "Record|Nomination
|Acquisition|Mensuration|GISRSTerminal   |Assessment    "

        Do Until rsNominations.EOF
            rsTargets.MoveFirst
                Do Until rsTargets.EOF Or bFound = True
                    If rsTargets!Nomination = rsNominations!Nomination
Then
                        bFound = True
                    Else
                        rsTargets.MoveNext
                    End If

                Loop
                With rsNominations

                    If bFound = False Then
                        FlexOutput.AddItem iRecord & vbTab &
!Nomination & vbTab & _
                                !Acquisition & vbTab & !Mensuration & vbTab
& !GISRSTerminal & _
                                vbTab & !Assessment
                                iRecord = iRecord + 1
                    End If
                End With
            bFound = False

            rsNominations.MoveNext
        Loop

    End If


    'Targets fired upon (impacts)
    If frmQueries.optQuery(9).Value = True Then
        stSQL = "Select Impact.Impact, Target.TargetId,
Target.Description, " & _
                "FireCommand.Platform, Fire.RoundsFired, Fire.FireTime,
" & _
```

```
                "Impact.ImpactTime, Impact.BDA " & _
                "From Impact, Fire, FireCommand, Target " & _
                "Where Fire.Fire = Impact.FireEvent " & _
                "And FireCommand.FireCommand = Fire.FireCommand " & _
                "And Target.TargetId = FireCommand.TargetId"
        Set RS = db.OpenRecordset(stSQL)

        FlexOutput.FormatString = "Record|Impact |Target Id|Description
|" & _
                                  "Firer Platform   |Rounds|Fire Time
|" & _
                                  "Impact Time      |BDA
"

        iRecord = 1

        Do Until RS.EOF
            FlexOutput.AddItem iRecord & vbTab & RS!Impact & vbTab & _
                RS!TargetId & vbTab & RS!Description & vbTab &
RS!Platform & _
                vbTab & RS!RoundsFired & vbTab & RS!FireTime & vbTab &
_
                RS!ImpactTime & vbTab & RS!BDA
            RS.MoveNext
            iRecord = iRecord + 1

        Loop
    End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmQueries.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmQueries.Enabled = True
    Unload Me

End Sub


'************************************************************
'Module:        frmQuestions.frm
'Description:   Allows user to access the questions
'               records for addition, deletion, and
'               modification.
'Programmer:    Kevin Colón
'************************************************************
```

```
Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datQuestions.Recordset.AddNew
        txtQuestion.Enabled = True
        txtQuestion.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datQuestions.Enabled = False

    Else
        datQuestions.Recordset.CancelUpdate
        txtQuestion.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datQuestions.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping


End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datQuestions.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Question " & txtQuestion.Text & "?",
vbYesNo, "Delete Question")
        If iResp = vbYes Then
            With datQuestions.Recordset
```

```
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Question"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtQuestion <> "" And txtDescription <> "" Then
        txtQuestion = UCase(txtQuestion)
        iResp = MsgBox("Do you want to add " & txtQuestion & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Question")
        If iResp = vbYes Then
            datQuestions.Recordset.Update
        End If

    Else
        MsgBox "You must enter an Question and a description before
saving.", vbExclamation _
                , "Add Question"
        datQuestions.Recordset.CancelUpdate
    End If

    txtQuestion.Enabled = False
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
```

```
    datQuestions.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a Question name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datQuestions.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datQuestions.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtQuestion.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtQuestion.SetFocus
        cmdAdd.Enabled = False
        datQuestions.Enabled = False
        datQuestions.Recordset.Edit
    Else
        If datQuestions.Recordset.RecordCount > 0 Then
            datQuestions.Recordset.Update

            txtQuestion.Enabled = False
            txtDescription.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
```

```
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datQuestions.Enabled = True
        End If
    End If

End Sub

Private Sub datQuestions_Reposition()
    SetQuestionRecordNumber

End Sub

Private Sub Form_Load()
    datQuestions.DatabaseName = gstNewDatabase

    With datQuestions
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetQuestionRecordNumber

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datQuestions.Recordset.FindFirst "[Question] = '" & _
                InputBox("Enter the Question", "Question Search") & "'"

    If datQuestions.Recordset.NoMatch Then
        MsgBox "Question was not found.", vbOKOnly, "Question Search"
        datQuestions.Recordset.MoveFirst         'go to first record
    End If

End Sub
```

```
Private Sub SetQuestionRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datQuestions.Recordset.RecordCount
    iCurrentRecord = datQuestions.Recordset.AbsolutePosition + 1
    If datQuestions.Recordset.EOF Then
        datQuestions.Caption = "No more records"
    Else
        datQuestions.Caption = "Question " & iCurrentRecord & _
                               " of " & iRecordCount
    End If

End Sub


'************************************************************
'Module:         frmQuestions.frm
'Description:    Allows user to access the questions
'                records for addition, deletion, and
'                modification.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datQuestions.Recordset.AddNew
        txtQuestion.Enabled = True
        txtQuestion.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datQuestions.Enabled = False

    Else
        datQuestions.Recordset.CancelUpdate
        txtQuestion.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datQuestions.Enabled = True

    End If
```

```
cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping


End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datQuestions.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Question " & txtQuestion.Text & "?",
vbYesNo, "Delete Question")
        If iResp = vbYes Then
            With datQuestions.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Question"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub
```

```
Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtQuestion <> "" And txtDescription <> "" Then
        txtQuestion = UCase(txtQuestion)
        iResp = MsgBox("Do you want to add " & txtQuestion & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Question")
        If iResp = vbYes Then
            datQuestions.Recordset.Update
        End If

    Else
        MsgBox "You must enter an Question and a description before
saving.", vbExclamation _
                , "Add Question"
        datQuestions.Recordset.CancelUpdate
    End If

    txtQuestion.Enabled = False
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datQuestions.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a Question name before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datQuestions.Recordset.CancelUpdate
            Resume Next
    End Select
```

```
End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datQuestions.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtQuestion.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtQuestion.SetFocus
        cmdAdd.Enabled = False
        datQuestions.Enabled = False
        datQuestions.Recordset.Edit
    Else
        If datQuestions.Recordset.RecordCount > 0 Then
            datQuestions.Recordset.Update

            txtQuestion.Enabled = False
            txtDescription.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datQuestions.Enabled = True
        End If
    End If

End Sub

Private Sub datQuestions_Reposition()
    SetQuestionRecordNumber

End Sub

Private Sub Form_Load()
    datQuestions.DatabaseName = gstNewDatabase

    With datQuestions
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetQuestionRecordNumber
'

End Sub

Private Sub Form_Unload(Cancel As Integer)
```

```
    frmMain.Show
    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datQuestions.Recordset.FindFirst "[Question] = '" & _
                InputBox("Enter the Question", "Question Search") & "'"

    If datQuestions.Recordset.NoMatch Then
        MsgBox "Question was not found.", vbOKOnly, "Question Search"
        datQuestions.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetQuestionRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datQuestions.Recordset.RecordCount
    iCurrentRecord = datQuestions.Recordset.AbsolutePosition + 1
    If datQuestions.Recordset.EOF Then
        datQuestions.Caption = "No more records"
    Else
        datQuestions.Caption = "Question " & iCurrentRecord & _
                               " of " & iRecordCount
    End If

End Sub



'************************************************************
'Module:         frmSQL.frm
'Description:    Allows user to enter their own SQL statement
'                and provides the results in a data bound
'                control.
'Programmer:     Kevin Colón
'************************************************************

Option Explicit

Private Sub cmdCancel_Click()

    frmMain.Enabled = True
```

```
    Unload Me

End Sub

Private Sub cmdSearch_Click()
    Dim stMsg   As String


    On Error GoTo HandleQueryError

    datSQL.DatabaseName = gstNewDatabase

    datSQL.RecordSource = txtSQL
    datSQL.Refresh

cmdSearch_Click_Exit:
    Exit Sub


HandleQueryError:
    Select Case Err.Number
        Case 3078
            stMsg = "A table you entered is not recognized.  Please
verifythat this table exists."
            MsgBox stMsg, vbOKOnly, "Custom Query Error"

            txtSQL.SetFocus
            Exit Sub
    End Select


End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

'*************************************************************
'Module:         frmTargetEvent.frm
'Description:    Allows user to access the target records
'                for addition, deletion, and modification.
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit

Dim rsNomination    As Recordset
Dim rsWeaponType    As Recordset
Dim stSQL1          As String
Dim stSQL2          As String
Private WordApp     As Word.Application
Private Doc         As Word.Document
```

```
Private Sel         As Word.Selection


Private Sub cboNomination_Change()

    If cboNomination.ListIndex >= 0 Then
        txtNomination = cboNomination.Text
    End If

End Sub

Private Sub cboWeaponType_Change()

    If cboWeaponType.ListIndex >= 0 Then
        txtWeaponType = cboWeaponType.Text
    End If

End Sub

Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then

        datTarget.Recordset.AddNew
        cboNomination.Enabled = True
        cboWeaponType.Enabled = True
        cboNomination.ListIndex = -1
        cboWeaponType.ListIndex = -1
        txtTimeofDesignation.Enabled = True
        txtNLTTime.Enabled = True
        txtDescription.Enabled = True
        txtLocation.Enabled = True
        txtAltitude.Enabled = True
        txtSpeed.Enabled = True
        txtRemark.Enabled = True
        txtTargetId.Enabled = True
        txtPriority.Enabled = True
        txtDesiredEffect.Enabled = True
        cmdUpdate.Enabled = False
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdAdd.Caption = "&Cancel"
        mnuFile.Enabled = False
        datTarget.Enabled = False

    Else

        datTarget.Recordset.CancelUpdate
        cboNomination.Enabled = False
        cboWeaponType.Enabled = False
        txtTimeofDesignation.Enabled = False
        txtNLTTime.Enabled = False
        txtDescription.Enabled = False
```

```
        txtLocation.Enabled = False
        txtAltitude.Enabled = False
        txtSpeed.Enabled = False
        txtRemark.Enabled = False
        txtTargetId.Enabled = False
        txtPriority.Enabled = False
        txtDesiredEffect.Enabled = False
        cmdUpdate.Enabled = True
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdAdd.Caption = "&Add"
        mnuFile.Enabled = True
        datTarget.Enabled = True
        cmdAdd.SetFocus

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess      As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0     'turn off error trapping


End Sub

Private Sub cmdDel_Click()

    Dim iResp       As Integer

    On Error GoTo HandleDelErrors

    If datTarget.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Target " & txtTargetId & "?", vbYesNo, _
                "Delete Target")
        If iResp = vbYes Then
            With datTarget.Recordset
                .Delete
                .MoveNext
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation, "Delete Target"
```

```
    End If

cmdDel_Click:
    Exit Sub

HandleDelErrors:
    Dim stMess       As String

    stMess = "Cannot complete operation." & vbCrLf & vbCrLf &
Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0



End Sub

Private Sub cmdSave_Click()

    'save current record
    On Error GoTo HandleSaveErrors

    If cboNomination.ListIndex >= 0 And cboWeaponType.ListIndex >= 0
Then

        datTarget.Recordset.Update
    Else
        MsgBox "You must select a Nomination Event and a Weapon Type
before saving." _
                , vbExclamation, "Add Target Event"
        datTarget.Recordset.CancelUpdate
    End If

    cboNomination.Enabled = False
    cboWeaponType.Enabled = False
    txtTimeofDesignation.Enabled = False
    txtNLTTime.Enabled = False
    txtDescription.Enabled = False
    txtLocation.Enabled = False
    txtAltitude.Enabled = False
    txtSpeed.Enabled = False
    txtRemark.Enabled = False
    txtTargetId.Enabled = False
    txtPriority.Enabled = False
    txtDesiredEffect.Enabled = False
    cmdUpdate.Enabled = True
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    cmdAdd.Caption = "&Add"
    mnuFile.Enabled = True
    datTarget.Enabled = True
    cmdAdd.SetFocus


    datTarget.Enabled = True
```

```
cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Select Nomination Event and Weapon Type before
saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datTarget.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()

    If cmdUpdate.Caption = "&Update" And _
        datTarget.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        cboNomination.Enabled = True
        cboWeaponType.Enabled = True
        txtTimeofDesignation.Enabled = True
        txtNLTTime.Enabled = True
        txtDescription.Enabled = True
        txtLocation.Enabled = True
        txtAltitude.Enabled = True
        txtSpeed.Enabled = True
        txtRemark.Enabled = True
        txtTargetId.Enabled = True
        txtPriority.Enabled = True
        txtDesiredEffect.Enabled = True
        cmdAdd.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        datTarget.Enabled = False
        datTarget.Recordset.Edit

    Else
        If datTarget.Recordset.RecordCount > 0 Then
```

```
            datTarget.Recordset.Update

            cboNomination.Enabled = False
            cboWeaponType.Enabled = False
            txtTimeofDesignation.Enabled = False
            txtNLTTime.Enabled = False
            txtDescription.Enabled = False
            txtLocation.Enabled = False
            txtAltitude.Enabled = False
            txtSpeed.Enabled = False
            txtRemark.Enabled = False
            txtTargetId.Enabled = False
            txtPriority.Enabled = False
            txtDesiredEffect.Enabled = False
            cmdDel.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            mnuFile.Enabled = True
            datTarget.Enabled = True


        End If
    End If

End Sub

Private Sub datTarget_Reposition()

    SetTargetRecordNumber

End Sub

Private Sub Form_Load()


    datTarget.DatabaseName = gstNewDatabase '

    stSQL1 = "Select Nomination from Nomination"
    stSQL2 = "Select WeaponType from WeaponType"


    Set rsNomination = db.OpenRecordset(stSQL1)
    Set rsWeaponType = db.OpenRecordset(stSQL2)

    'fill cboNomination
    Do Until rsNomination.EOF
        cboNomination.AddItem rsNomination!Nomination
        rsNomination.MoveNext
    Loop

    'fill cboWeaponType
    Do Until rsWeaponType.EOF
        cboWeaponType.AddItem rsWeaponType!WeaponType
        rsWeaponType.MoveNext
```

```
    Loop

    With datTarget
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetTargetRecordNumber

End Sub

Private Sub SetTargetRecordNumber()

    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datTarget.Recordset.RecordCount
    iCurrentRecord = datTarget.Recordset.AbsolutePosition + 1

    If datTarget.Recordset.EOF Then
        datTarget.Caption = "No more records"
    Else
        datTarget.Caption = "Target Record " & iCurrentRecord & _
                            " of " & iRecordCount
    End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFilePrint_Click()
    frmPrint.Show

    On Error GoTo mnuPrintErrors

    If bContinue = True Then

        With datTarget.Recordset

            If bWord = True Then
```

```
                Set WordApp = New Word.Application
                WordApp.Documents.Add
                Set Doc = WordApp.ActiveDocument
                Set Sel = WordApp.Selection

                Doc.Tables.Add Range:=Sel.Range, NumRows:=.RecordCount,
NumColumns:=8


                Sel.TypeText Text:="TargetId"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Designation Time"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Nomination"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Location"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Altitude"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Speed"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="NLT Time"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Priority"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Weapon Type"
                Sel.MoveRight unit:=12                  '12=next cell

                Sel.TypeText Text:="Remark"
                Sel.MoveRight unit:=12                  '12=next cell

                Do Until .EOF

                    Sel.TypeText Text:=!TargetId
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TimeofDesignation
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!Nomination
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TargetLocation
                    Sel.MoveRight unit:=12                  '12=next
```

```
cell

                    Sel.TypeText Text:=!TargetAltitude
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TargetSpeed
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!TargetNLTTime
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!Priority
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!WeaponType
                    Sel.MoveRight unit:=12                  '12=next
cell

                    Sel.TypeText Text:=!Remark
                    Sel.MoveRight unit:=12                  '12=next
cell

                    .MoveNext

                Loop

                WordApp.Visible = True

                Set WordApp = Nothing


            Else
                If bText = True Then

                    Open App.Path & "\NomEvents.txt" For Output As #1

                    Print #1, "TargetId"; Chr(9); "Designation Time";
Chr(9); "Nomination"; Chr(9); _
                                "Location"; Chr(9); "Altitude"; Chr(9);
_
                                "Speed"; Chr(9); "NLTTime"; Chr(9); _
                                "Priority"; Chr(9); "WeaponType";
Chr(9); _
                                "Remark"; Chr(9)

                    Do Until .EOF

                        Print #1, !TargetId; Chr(9); _
                                  !TimeofDesignation; Chr(9); _
                                  !Nomination; Chr(9); _
                                  !TargetLocation; Chr(9); _
```

```
                                    !TargetAltitude; Chr(9); _
                                    !TargetSpeed; Chr(9); _
                                    !TargetNLTTime; Chr(9); _
                                    !Priority; Chr(9); _
                                    !WeaponType; Chr(9); _
                                    !Remark; Chr(9)

                        .MoveNext
                    Loop

                    Close #1

                End If
            End If

        .MoveFirst

        End With

    End If

    bContinue = False
    bWord = False
    bText = False


mnuPrintErrors:
        Select Case Err.Number
            Case 94
                Sel.TypeText Text:=""
                Resume Next
        End Select

End Sub

Private Sub txtNomination_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsNomination.MoveFirst
    If txtNomination <> "" Then
        Do Until iIndex = rsNomination.RecordCount Or bFound
            If rsNomination!Nomination = txtNomination Then
                cboNomination.Text = rsNomination!Nomination
                bFound = True
            Else
                rsNomination.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If
```

```
End Sub

Private Sub txtWeaponType_Change()

    'selects correct combo box listing
    Dim iIndex      As Integer
    Dim bFound      As Boolean

    rsWeaponType.MoveFirst
    If txtWeaponType <> "" Then
        Do Until iIndex = rsWeaponType.RecordCount Or bFound
            If rsWeaponType!WeaponType = txtWeaponType Then
                cboWeaponType.Text = rsWeaponType!WeaponType
                bFound = True
            Else
                rsWeaponType.MoveNext
                iIndex = iIndex + 1
            End If

        Loop
    End If

End Sub



'*****************************************************************
'Module:         frmTargets2.frm
'Description:    Allows user to view all target records.  Uses
'                the filters form to reduce the number of
'                records displayed.
'Programmer:     Kevin Colón
'*****************************************************************

Option Explicit


Private Sub FlexTargets_DblClick()

    frmTimeline.Show

End Sub

Private Sub dbgTargets_DblClick()

    frmTimeline.Show

End Sub

Private Sub Form_Load()
    Dim stSQL As String
    Dim iRecord As Integer

    stSQL = "Select * from Target"
```

```
    datTargets.DatabaseName = gstNewDatabase

    datTargets.RecordSource = stSQL
    datTargets.Refresh


End Sub

Private Sub Form_Resize()

    dbgTargets.Width = Me.Width

End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFilters_Click()

    frmFilters.Show
    Me.Enabled = False


End Sub



'*************************************************************
'Module:        frmThreatTypes.frm
'Description:   Allows user to access the threat types
'               records for addition, deletion, and
'               modification.
'Programmer:    Kevin Colón
'*************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datThreatTypes.Recordset.AddNew
        txtThreatType.Enabled = True
        txtThreatType.SetFocus
```

```
        txtDescription.Enabled = True
        txtMission.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datThreatTypes.Enabled = False

    Else
        datThreatTypes.Recordset.CancelUpdate
        txtThreatType.Enabled = False
        txtDescription.Enabled = False
        txtMission.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datThreatTypes.Enabled = True

    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datThreatTypes.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Threat " & txtThreatType.Text & "?",
vbYesNo, "Delete Threat")
        If iResp = vbYes Then
            With datThreatTypes.Recordset
                .Delete         'delete current record
                .MoveNext       'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
```

```
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Threat"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0         'turn off error trapping

End Sub

Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtThreatType.Text <> "" Then
        txtThreatType.Text = UCase(txtThreatType.Text)
        iResp = MsgBox("Do you want to add " & txtThreatType.Text & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Threat")
        If iResp = vbYes Then
            datThreatTypes.Recordset.Update
        End If

    Else
        MsgBox "You must enter a Threat type before saving.",
vbExclamation _
                , "Add Threat"
        datThreatTypes.Recordset.CancelUpdate
    End If

    txtThreatType.Enabled = False
    txtDescription.Enabled = False
    txtMission.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datThreatTypes.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True
```

```
cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a Threat type before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datThreatTypes.Recordset.CancelUpdate
            Resume Next
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datThreatTypes.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtThreatType.Enabled = True
        txtDescription.Enabled = True
        txtMission.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtThreatType.SetFocus
        cmdAdd.Enabled = False
        datThreatTypes.Enabled = False
        datThreatTypes.Recordset.Edit
    Else
        If datThreatTypes.Recordset.RecordCount > 0 Then
            datThreatTypes.Recordset.Update

            txtThreatType.Enabled = False
            txtDescription.Enabled = False
            txtMission.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datThreatTypes.Enabled = True
        End If
    End If
```

```
End Sub

Private Sub datThreatTypes_Reposition()

    SetThreatRecordNumber

End Sub

Private Sub Form_Load()

    datThreatTypes.DatabaseName = gstNewDatabase

    With datThreatTypes
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetThreatRecordNumber


End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datThreatTypes.Recordset.FindFirst "[ThreatType] = '" & _
                InputBox("Enter the Threat Type", "Threat Type Search")
& "'"

    If datThreatTypes.Recordset.NoMatch Then
        MsgBox "Threat Type was not found.", vbOKOnly, "Threat Type
Search"
        datThreatTypes.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetThreatRecordNumber()
```

```
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datThreatTypes.Recordset.RecordCount
    iCurrentRecord = datThreatTypes.Recordset.AbsolutePosition + 1
    If datThreatTypes.Recordset.EOF Then
        datThreatTypes.Caption = "No more records"
    Else
        datThreatTypes.Caption = "Threat " & iCurrentRecord & _
                                 " of " & iRecordCount
    End If

End Sub



'*************************************************************
'Module:         frmTimeline.frm
'Description:    Displays the event timeline from acquisition
'                to impact for a selected target
'Programmer:     Kevin Colón
'*************************************************************

Option Explicit


Private Sub Form_Load()
    Dim stSQL As String
    Dim stTarget As String
    Dim iTabs As Integer
    Dim rsTimes As Recordset
    Dim stSQL1 As String
    Dim stSQL2 As String
    Dim stSQL3 As String
    Dim stSQL4 As String
    Dim stSQL5 As String
    Dim stSQL6 As String
    Dim stSQL7 As String
    Dim rsAcquisition As Recordset
    Dim rsMensuration As Recordset
    Dim rsNomination As Recordset
    Dim rsTarget As Recordset
    Dim rsFireCommand As Recordset
    Dim rsFire As Recordset
    Dim rsImpact As Recordset

    frmTargets2.dbgTargets.Col = 0
    stTarget = frmTargets2.dbgTargets.Text

    stSQL4 = "Select * from Target where Target.TargetId = '" &
stTarget & "'"
    Set rsTarget = db.OpenRecordset(stSQL4)

    stSQL3 = "Select * from Nomination where Nomination.Nomination = '"
```

```
& rsTarget!Nomination & "'"
    Set rsNomination = db.OpenRecordset(stSQL3)

    stSQL2 = "Select * from Mensuration where Mensuration.Mensuration =
'" & rsNomination!Mensuration & "'"
    Set rsMensuration = db.OpenRecordset(stSQL2)

    stSQL1 = "Select * from Acquisition where Acquisition.Acquisition =
'" & rsNomination!Acquisition & "'"
    Set rsAcquisition = db.OpenRecordset(stSQL1)

    stSQL5 = "Select * from FireCommand where FireCommand.TargetId = '"
& stTarget & "'"
    Set rsFireCommand = db.OpenRecordset(stSQL5)

    stSQL6 = "Select * from Fire where Fire.FireCommand = '" &
rsFireCommand!FireCommand & "'"
    Set rsFire = db.OpenRecordset(stSQL6)

    If rsFire.RecordCount > 0 Then
        stSQL7 = "Select * from Impact where Impact.FireEvent = '" &
rsFire!Fire & "'"
        Set rsImpact = db.OpenRecordset(stSQL7)
    End If


    lstTimeline.AddItem "Target Id: " & stTarget
    lstTimeline.AddItem "Target Description:    " &
rsTarget!Description
    lstTimeline.AddItem ""
    lstTimeline.AddItem "NLT Time:                          " &
rsTarget!TargetNLTTime
    lstTimeline.AddItem ""
    lstTimeline.AddItem "Acquisition Time:                " &
rsAcquisition!AcqTime
    lstTimeline.AddItem "Mensuration Rqst:              " &
rsMensuration!TimeRequestSent
    lstTimeline.AddItem "Mensuration Rcvd:             " &
rsMensuration!TimeRequestReceived
    lstTimeline.AddItem "Mensuration Info Sent:       " &
rsMensuration!TimeRequestReceived
    lstTimeline.AddItem "Mensuration Info Rcvd:      " &
rsMensuration!TimeRequestReceived
    lstTimeline.AddItem "Nomination Sent:               " &
rsNomination!NomTimeSent
    lstTimeline.AddItem "Nomination Rcvd:              " &
rsNomination!NomTimeRcvd
    lstTimeline.AddItem "Target Designation:           " &
rsTarget!TimeofDesignation
    lstTimeline.AddItem "Fire Command Xmit:           " &
rsFireCommand!FCTimeXmit
    lstTimeline.AddItem "Fire Command Rcvd:          " &
rsFireCommand!FCTimeRcvd

    If rsFire.RecordCount > 0 Then
```

```
        lstTimeline.AddItem "Fire Event: " & rsFire!FireTime
        If rsImpact.RecordCount > 0 Then
            lstTimeline.AddItem "Impact: " & rsImpact!ImpactTime
        End If
    End If

    frmTargets2.Enabled = False


End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmTargets2.Enabled = True
    Unload Me

End Sub

'************************************************************
'Module:        frmWeaponTypes.frm
'Description:   Allows user to access the weapon types
'               records for addition, deletion, and
'               modification.
'Programmer:    Kevin Colón
'************************************************************

Option Explicit

Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors

    If cmdAdd.Caption = "&Add" Then
        datWeaponTypes.Recordset.AddNew
        txtWeaponType.Enabled = True
        txtWeaponType.SetFocus
        txtDescription.Enabled = True
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
        cmdDel.Enabled = False
        cmdUpdate.Enabled = False
        mnuFile.Enabled = False
        datWeaponTypes.Enabled = False

    Else
        datWeaponTypes.Recordset.CancelUpdate
        txtWeaponType.Enabled = False
        txtDescription.Enabled = False
        cmdSave.Enabled = False
        cmdDel.Enabled = True
        cmdUpdate.Enabled = True
        mnuFile.Enabled = True
        cmdAdd.Caption = "&Add"
        cmdAdd.SetFocus
        datWeaponTypes.Enabled = True
```

```
    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim stMess  As String
    stMess = "Cannot complete operation. " & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMess, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub

Private Sub cmdDel_Click()
    'delete the current record
    Dim iResp   As Integer

    On Error GoTo HandleDelErrors

    If datWeaponTypes.Recordset.RecordCount > 0 Then
        iResp = MsgBox("Delete Weapon " & txtWeaponType.Text & "?",
vbYesNo, "Delete Weapon")
        If iResp = vbYes Then
            With datWeaponTypes.Recordset
                .Delete          'delete current record
                .MoveNext        'move to following record
                If .EOF Then
                    .MovePrevious
                    If .BOF Then
                        MsgBox "The recordset is empty.",
vbInformation, "No Records"
                    End If
                End If
            End With
        End If
    Else
        MsgBox "No records to delete.", vbExclamation _
                , "Delete Weapon"

    End If

cmdDel_Click_Exit:
    Exit Sub

HandleDelErrors:
    Dim stMsg   As String

    stMsg = "Cannot complete operation." & vbCrLf & vbCrLf _
            & Err.Description
    MsgBox stMsg, vbExclamation, "Database Error"
    On Error GoTo 0          'turn off error trapping

End Sub
```

```
Private Sub cmdSave_Click()
    'save the current record
    Dim iResp   As Integer

    On Error GoTo HandleSaveErrors
    If txtWeaponType.Text <> "" Then
        txtWeaponType.Text = UCase(txtWeaponType.Text)
        iResp = MsgBox("Do you want to add " & txtWeaponType.Text & _
                    " to the database?", vbYesNo + vbQuestion, _
                    "Add Weapon")
        If iResp = vbYes Then
            datWeaponTypes.Recordset.Update
        End If

    Else
        MsgBox "You must enter a weapon type before saving.",
vbExclamation _
                , "Add Weapon"
        datWeaponTypes.Recordset.CancelUpdate
    End If

    txtWeaponType.Enabled = False
    txtDescription.Enabled = False
    cmdSave.Enabled = False
    cmdDel.Enabled = True
    datWeaponTypes.Enabled = True
    mnuFile.Enabled = True
    cmdAdd.Caption = "&Add"
    cmdAdd.SetFocus
    cmdUpdate.Enabled = True


cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim stMess  As String
    Select Case Err.Number
        Case 3022           'duplicate key field
            stMess = "Record already exists -- could not save>'"
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case 3058, 3315     'no entry in key field
            stMess = "Enter a weapon type before saving."
            MsgBox stMess, vbExclamation, "Database Error"
            On Error GoTo 0     'turn off error trapping

        Case Else
            stMess = "Record could not be saved." & vbCrLf _
                    & Err.Description
            MsgBox stMess, vbExclamation, "Database Error"
            datWeaponTypes.Recordset.CancelUpdate
            Resume Next
```

```
    End Select

End Sub

Private Sub cmdUpdate_Click()
    If cmdUpdate.Caption = "&Update" And _
        datWeaponTypes.Recordset.RecordCount > 0 Then

        cmdUpdate.Caption = "Su&bmit"
        txtWeaponType.Enabled = True
        txtDescription.Enabled = True
        cmdDel.Enabled = False
        mnuFile.Enabled = False
        txtWeaponType.SetFocus
        cmdAdd.Enabled = False
        datWeaponTypes.Enabled = False
        datWeaponTypes.Recordset.Edit
    Else
        If datWeaponTypes.Recordset.RecordCount > 0 Then
            datWeaponTypes.Recordset.Update

            txtWeaponType.Enabled = False
            txtDescription.Enabled = False
            cmdDel.Enabled = True
            mnuFile.Enabled = True
            cmdAdd.Enabled = True
            cmdAdd.SetFocus
            cmdUpdate.Caption = "&Update"
            datWeaponTypes.Enabled = True
        End If
    End If

End Sub

Private Sub datWeaponTypes_Reposition()

    SetWeaponRecordNumber

End Sub

Private Sub Form_Load()

    datWeaponTypes.DatabaseName = gstNewDatabase

    With datWeaponTypes
        .Refresh
        If Not .Recordset.EOF Then
            .Recordset.MoveLast
            .Recordset.MoveFirst
        End If
    End With

    SetWeaponRecordNumber
```

```
End Sub

Private Sub Form_Unload(Cancel As Integer)

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileBack_Click()

    frmMain.Enabled = True
    Unload Me

End Sub

Private Sub mnuFileSearch_Click()

    datWeaponTypes.Recordset.FindFirst "[WeaponType] = '" & _
                InputBox("Enter the Weapon Type", "Weapon Type Search")
& "'"

    If datWeaponTypes.Recordset.NoMatch Then
        MsgBox "Weapon Type was not found.", vbOKOnly, "Weapon Type
Search"
        datWeaponTypes.Recordset.MoveFirst         'go to first record
    End If

End Sub

Private Sub SetWeaponRecordNumber()
    Dim iRecordCount    As Integer
    Dim iCurrentRecord  As Integer

    iRecordCount = datWeaponTypes.Recordset.RecordCount
    iCurrentRecord = datWeaponTypes.Recordset.AbsolutePosition + 1
    If datWeaponTypes.Recordset.EOF Then
        datWeaponTypes.Caption = "No more records"
    Else
        datWeaponTypes.Caption = "Weapon " & iCurrentRecord & _
                            " of " & iRecordCount
    End If

End Sub
```

LIST OF REFERENCES

1. Boggs, M. and Boggs, W., *Mastering UML with Rational Rose*, Sybex, 1999.

2. Jennings, R., *Using Access 97 Second Edition*, Que, 1997.

3. Levin, S., GDIS, "Land Attack Warfare System Employment in FBE Golf ." 24 May 2000.

4. McManus, J.P., *Database Access with Visual Basic*, pp. 262-265, Sams, 1998.

5. Naval Postgraduate School, IJWA, *Fleet Battle Experiment Echo Data Capture and Analysis,* 9 March 1999.

6. Naval Postgraduate School, IJWA, *Fleet Battle Experiment Echo Data Capture and Analysis Report*, 1 October 1999.

7. Naval Postgraduate School, IJWA, *Fleet Battle Experiment Foxtrot Experiment Plan*, 23 Nov 1999.

8. Naval Postgraduate School, IJWA, *Fleet Battle Experiment Golf: Appendix E*, pp. 1-21, March-April 2000.

9. Naval Postgraduate School, IJWA, *Time Critical Targeting Concept of Operations (CONOPS) for Fleet Battle Experiment Golf: MBC-Final Draft*, pp. 1-89, 2 March 2000.

10. NWC Public Affairs, "New Thinking, New Building, Showcased at Last Naval War College Wargame of Century." [http://www.nwdc.navy.mil]. September 1999.

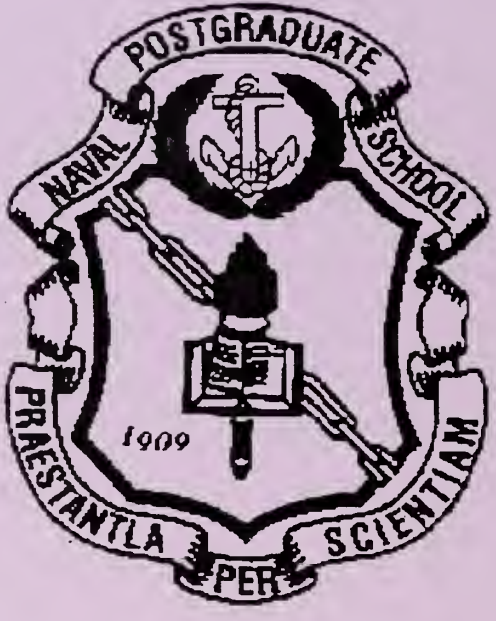
11. NWC Public Affairs, "Maritime Battle Center." [http://www.nwdc.navy.mil/navagation/mbc.htm]. Date not available.

12. Tracy, P.A. (VADM), "Military Education for 21st Century Warrior," [http://web.nps.navy.mil/FutureWarrior/Presentations/Tracey/tsld001.htm.]. 1997.

THIS PAGE INTENTIONALLY LEFT BLANK

Distribution List

Defense Technical Information Center	2
Library, Naval Postgraduate School	2
Gordon Schacher, IJWA, NPS	10
William Kemple, IJWA, NPS	1
Phil Depoy, IJWA, NPS	1
Shelley Gallup, IJWA, NPS	1
Chuck Marashian, IJWA, NPS	1
Cathy Spencer, IJWA, NPS	1
Rich Kimmel, IJWA, NPS	1
Magdi Kamel	1
Bob VanZandt, J9, JFCOM	5
COL Chris Shepherd, J9, JFCOM	1
Raquelle Hill, J9, JFCOM	1



Naval Postgraduate School
Monterey, California